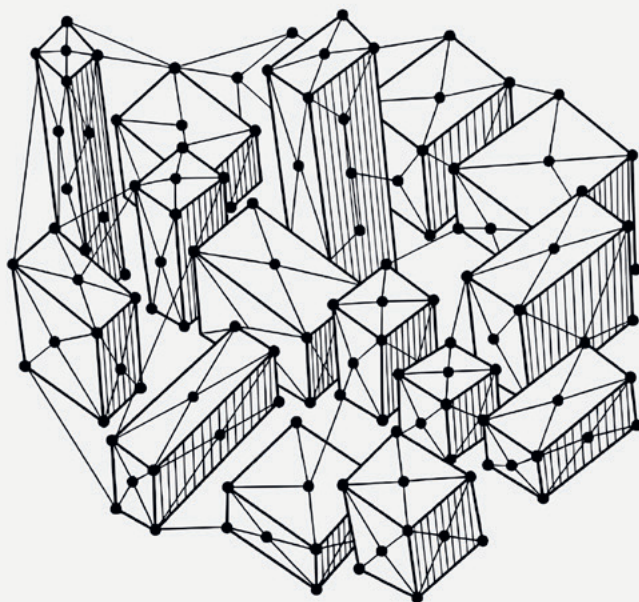


Buildings as Networks

Modelling Built Heritage Knowledge Through Graphs

Angelo Massafra



Bologna
University Press



ALMA MATER STUDIORUM | DIPARTIMENTO
UNIVERSITÀ DI BOLOGNA | DI ARCHITETTURA

Collana del Dipartimento di Architettura – DA
Dottorato in Architettura e Culture del Progetto
Alma Mater Studiorum – Università di Bologna
Direttore del Dipartimento: Fabrizio Ivan Apollonio

Comitato Scientifico:

Cristina Gonzalez Longo, Carola Hein, Piotr Kuroczynski, Dijon Moraes jr., Ana C. Mourao Moura,
Henrik Reeh, Rosa Schiano-Phan, Uwe Schroeder

Comitato di Indirizzo: Collegio del Dottorato in Architettura e Culture del Progetto

Tutti i contributi pubblicati all'interno della Collana sono sottoposti a un processo di *double-blind peer review*.



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO
DI ARCHITETTURA



Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PILLOLE DI RICERCA E INNOVAZIONE

The contents of this study are the result of competitive research projects in the field of building management carried out at the Department of Architecture of the University of Bologna. In particular, this work was developed within the framework of the following funded projects: the European Horizon Europe project "HERIT4AGES: User-centric and data-driven retrofitting solutions for a resilient, energy-efficient, low-emission and inclusive cultural heritage" (Grant Agreement No. 101123175); the Italian PRIN 2022 project "Occupant-based DIGITal predictive MANagement to improve the built environment – DIGITMAN" (Project Code 2022JMRXZA_002, CUP J53D23002180006), funded under the National Recovery and Resilience Plan (PNRR), Mission 4, Component 2, Investment 1.1, "Fondo per il Programma Nazionale di Ricerca e Progetti di Rilevante Interesse Nazionale (PRIN)" (Call issued by the Italian Ministry of University and Research, Ministerial Decree No. 104 of 02/02/2022); and the University of Bologna ALMAVALUE project "BeTwin: Building Digital Twins for Built Heritage Performance-Based Management". The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the funding bodies, nor can the funding bodies be held responsible for them.

Fondazione Bologna University Press
Via Saragozza 10, 40123 Bologna
tel. (+39) 051 232 882
ISSN 2385-0515
SBN: 979-12-5477-794-7
ISBN online: 979-12-5477-795-4
Doi: 10.30682/9791254777954

Copyright © Authors 2026
CC BY 4.0 License

www.buponline.com
info@buponline.com

Questo volume è stato realizzato a partire da un impaginato camera-ready fornito dall'Autore.

In copertina: immagine realizzata dall'Autore
Progetto grafico: Gianluca Bollina-DoppioClickArt (Bologna)
Prima edizione: marzo 2026

Buildings as Networks

Modelling Built Heritage Knowledge Through Graphs

Angelo Massafra

SUMMARY

7	<i>Foreword</i>
	Riccardo Gulli
11	<i>Introduction</i>
CONCEPTUAL AND TECHNOLOGICAL FOUNDATIONS	
19	1. <i>The emergence of graphs in science and technology</i>
43	2. <i>Graphs in the AECO domain</i>
METHODOLOGICAL FRAMEWORK	
71	3. <i>Federating ontologies for cross-disciplinary knowledge modelling</i>
105	4. <i>Operationalising graph-based workflows</i>
APPLICATIONS AND CASE STUDIES	
125	5. <i>Spatial graphs for the analysis of architectural layouts</i>
149	6. <i>Topological graphs for safety evaluation of evacuation networks</i>
181	7. <i>Semantic graphs for data integration in heritage building characterisation processes</i>
203	8. <i>Performance graphs for decision support in energy management</i>
231	9. <i>Scenario graphs for multicriteria decision making</i>
FUTURE SCENARIOS	
273	10. <i>Towards AI-integrated knowledge systems</i>
305	<i>References</i>
323	<i>Glossary</i>

FOREWORD

*Riccardo Gulli**

This contribution is part of the research activities of the Architectural Technology group at the Department of Architecture in Bologna, under my supervision. It is a scientific domain that brings together two worlds: the first concerns knowledge in the field of building theory and practice; the second concerns digital technologies, foremost among them Artificial Intelligence. The hinge between these two worlds is represented by the renewed epistemological framework induced by the need to ‘teach machines.’

This new domain of knowledge is at the centre of an epochal transition affecting all sectors of knowledge and all human activities, including the building sector. The focal point of this paradigm shift – typical of all scientific revolutions – does not lie solely in the technological dimension of the problem, but rather in the more complex issue of awareness in decision making, from which consequently derives the question of the ethics of responsibility.

It is now well established that the new frontiers of Artificial Intelligence foreshadow a radical transformation of professional roles. In the field of architectural design, for example, the authorship of an idea is progressively losing its relevance, especially in the concept design phase. Even more evident and pronounced is this shift in the field of structural engineering, where the professional role already today largely coincides with verifying the correctness of outputs produced by a computer, an activity that tomorrow may itself be easily replaced, with a level of reliability likely to be higher.

If we therefore consider it inevitable that this will occur in the near future – indeed within just a few years – the main issue on which attention should currently be focused concerns a simple question: once the tools and methods to transfer to a machine a body of knowledge as extensive as that of humans

**Department of
Architecture, Alma Mater
Studiorum - University of
Bologna, Italy*

have been defined, how can we maintain a degree of control that allows, on the one hand, education of the contents and, on the other, validation of the outcomes generated by a machine?

This question, in its generality, implies deploying a different approach to knowledge both in university education and in research. The core of this new approach lies in identifying the conceptual structures that characterise a specific disciplinary field, providing specific answers to specific questions, which in this context are represented by the “prompt.” This field of activity can be labelled “prompt engineering,” that is, the generation of conceptual maps for the formalisation of a problem. Following the transformation brought about by the rise of generative Artificial Intelligence, this field is no longer the exclusive domain of computer science. The possibility now available to bypass the use of computer code for the mathematical formalisation of a problem indeed urges a substantial revision of the spectrum of disciplinary competencies, as every human activity is increasingly permeated by digital technologies.

The work of Angelo Massafra presented here is an expression of this direction. The focus of the discussion is the graph, understood here as a conceptual structure formed by a system of relationships between the entities and properties of an object, which in computer science is known as an ontology. This form of knowledge also opens to the extended dimension of network data (Graph Data Science) in order to extract and manage informational content according to the logical structures of the graph. But the key step lies in connecting theory and practice, namely, in understanding the effectiveness of this approach in solving applied problems related to the design, construction, and management of the built environment.

A crucial point in this process is the ability to identify the characteristics of a building. This is, in fact, a key step that then allows the activation of subsequent levels of analysis on specific thematic aspects, including performance-related ones. Identifying the characteristics of an object is a broadly applicable problem and, in the case of buildings, consists of two steps: the first concerns visible characteristics, which can be inferred directly from an image of the object; the second concerns hidden characteristics, which represent its typological and constructional nature.

The methods by which this recognition process is activated involve the formalisation of an archetype that summarises the object's identifying characteristics and allows the creation of sets – called clusters – representing certain invariants. The formation of these clusters is a human-guided process based on entering selected data about the object, derived from the interplay between its iconographic and textual components—in other words, between images, drawings, and technical information provided to the machine for many buildings. A second phase then follows, consisting of validating the machine-generated outputs to refine and calibrate the results. The third phase tests the machine's autonomous recognition capabilities by surveying building images available on Google Maps.

This process exemplifies an approach to knowledge that combines the machine's ability to manipulate and process large amounts of data with the disciplinary knowledge embedded in the human capacity to elaborate conceptual content, informed in part by lived experience. The hinge between these two epistemological dimensions is thus prompt. The formulation of a question expresses conceptual knowledge that arises from intentions unknown to the machine, except after a process of acquiring content provided by humans. This foreshadows an epochal transition that enables us to move beyond the use of computer code and activate a direct correlation between the conceptual formalisation of a problem and its translation into a workflow tool.

Angelo Massafra's work addresses these same objectives and stands as an important contribution to advancing research in this field.

INTRODUCTION

In recent years, the Architecture, Engineering, Construction, and Operation (AECO) sector has begun a digital transformation, driven by technologies such as Building Information Modelling (BIM), the Internet of Things (IoT), Building Performance Simulation (BPS), Digital Twins (DT), and Artificial Intelligence (AI) (Alves *et al.* 2025; Deng *et al.* 2021).

Despite this rapid technological development, the industry continues to face fundamental challenges in information management. A significant impediment consists of the usual fragmentation of building data across silos – diverse models, repositories, software platforms, and stakeholders – which leads to interoperability problems (Merino *et al.* 2023). The vast heterogeneity of data sources, rarely integrated into systemic visions, makes it challenging to transform raw data into valuable information and decision-supporting knowledge. As a consequence of data fragmentation, owners and facility managers, particularly in the public sector, struggle to use data effectively, hindering the inclusion of knowledge-driven approaches in building management (Abuimara *et al.* 2021a).

This problem is not merely technological, but also conceptual and cultural. Among the many reasons, it stems from the complexity of the sector, which makes it impossible to digitally represent the whole ‘domain of buildings’ through a single, shared data schema or semantic structure. Indeed, any attempt to force all building knowledge into a single, uniform formal representation would risk oversimplifying reality and undermining the richness of the built environment in the virtual world.

In response to these challenges, and to harness the opportunities offered by digitalisation, researchers and practitioners began in recent years to develop methods and tools for the digital modelling of building knowledge, considering the need for conceptualising systemic frameworks as an essential prereq-

uisite for creating next-generation decision-support tools for the management of the built heritage (Petri *et al.* 2023; Succar and Poirier 2020; Zhang *et al.* 2025). However, this is a complex endeavour that rests on a non-trivial compromise: on the one hand, contemporary digital environments require information to be organised with sufficient ‘vertical’ precision to meet the specific needs of building management; on the other hand, they need knowledge structures that are ‘horizontally’ flexible enough to support a holistic approach in the construction industry.

This results in a pressing motivation to create knowledge-modelling frameworks that integrate diversified data - relative to different phases of the building lifecycle, spanning from design to operation and renovation, and to multiple subdomains of the AECO, like energy, sustainability, wellbeing, safety, etc. - into cohesive, queryable, and shareable knowledge bases. Such frameworks must promote semantic interoperability - i.e., different systems and players should be able to exchange and use the data - while accommodating the modularity of the domain. In other words, there is a need for modern knowledge modelling approaches, approaches that are holistic yet detailed, rigorous yet flexible, computable yet human-centred. Reconciling these seemingly paradoxical dichotomies demands a robust disciplinary apparatus in the field of building knowledge modelling, an apparatus to which this book seeks to contribute, at least in part, by conceptualising ‘*Buildings as Networks*’.

Graphs are data structures consisting of two simple types of elements: nodes and edges. They can be used to map both entities (i.e., the nodes) and the relationships between them (i.e., the edges) to represent a knowledge domain. The advantages offered by such knowledge structures are principally twofold. First, the natural capacity of graphs to encode connectivity and relationships, which is particularly valuable in complex systems such as buildings, where understanding interdependencies is as crucial as understanding individual components (Ghosh *et al.* 2023). Second, the possibility to enrich graph elements with semantic attributes, thus forming the so-called ‘knowledge graphs’ (KGs). These are graphs in which nodes and edges carry the explicit meanings defined by a formal schema - such as an ontology - that specifies the types of entities and relationships

that can exist within a domain, thus providing a shared vocabulary and framework for logical reasoning and inference (Ramonell *et al.* 2023).

By their nature, graphs can describe the world in a readily transferable way and pragmatically present humans' abstract ideas through clear visual models. In any domain, choosing appropriate techniques for constructing graphs is crucial, as different methodologies can yield distinct formal representations that convey the same knowledge in diverse ways. If a graph is too simplistic, it might fail to capture the complexity of a real-world issue; conversely, if it is too complex, it could hinder human comprehension.

The book is therefore based on the central assumption that a building – understood as an interconnected system of objects, spaces, and people – can be digitally represented as a network. To this end, focusing on building performance management, the discussion presents a modelling approach that combines multiple domain-specific ontologies into a single, unified knowledge framework capable of describing the many aspects that characterise *building performance* (energy, safety, environment, well-being, and so forth). This strategy, referred to as 'ontology federation', acknowledges that, rather than attempting to model the broad domain of building performance under a single, all-encompassing formal schema, it is preferable to structure a federated system in which small, specialised ontologies – each with its own semantic rules – contribute to representing the specific subdomain for which they were designed (Moretti *et al.* 2023).

This approach results in a federated KG that functions as a core integration layer, capable of linking diverse and separate data sources within a common formal representation, thereby enabling a systemic representation of building performance knowledge, interpretable by both humans and machines. The federated ontology ensures that each piece of data is contextualised, laying the groundwork for semantic interoperability between tools and stakeholders, allowing them to add or extract information from the same knowledge base without loss of meaning.

Implementing this approach required developing specific methods and tools, as introduced in the book. At a high level, the

methodology is operationalised through an open-source software toolkit that embodies the federated ontology within graph structures (Massafra *et al.* 2024a). The toolkit provides practical means to ingest data from various sources, map that data to the ontology, and store and query it as a KG. Its key design principle is a connector-based architecture, where lightweight connectors or adapters are provided for each data source (for example, BIM, sensor time-series databases, and BPS models). Each connector translates its native data into a representation that conforms to the KG. This allows each source to remain in its native format while being linked semantically to the central KG.

To validate the framework, the text presents a series of demonstrative case studies drawn from real-world scenarios. These applications were conducted within the Department of Architecture of the University of Bologna over the last three years, as part of multiple research initiatives, including the author's PhD in Architecture and Design Cultures, the BETWIN project (Building Digital Twins for Built Heritage Performance-Based Management, AlmaValue 2023 Call, 2023–2025), and the DIGITMAN project (Occupant-based DIGITal predictive MANagement to improve the built environment, PRIN 2022 Call, Prot. 2022JMRX2A). Additional explorations were carried out in collaboration with the Welsh School of Architecture, Cardiff University (UK), and the Dipartimento Unità Efficienza Energetica at ENEA (Agenzia Nazionale per le Nuove Tecnologie, l'Energia e lo Sviluppo Economico Sostenibile). These case studies apply the methodology to different facets of building performance, demonstrating how the approach can be generalised and reused across domains.

The applications are presented in order of increasing complexity, illustrating how graph-based methods range from the straightforward mapping of a connectivity system among spaces to the management of federated specialist models and the aggregation of performance data of various kinds across operational scenarios.

In brief, the demonstrations encompass the use of: *spatial graphs* for the analysis of residential building layouts, *topological graphs* for evaluating emergency egress routes in university facilities, *semantic graphs* for integrating heterogeneous data in heritage buildings' characterisation processes, *performance*

graphs for linking BIM and BEM models in energy analysis, and *temporal graphs* for scenario-based strategic management in higher education contexts. The presentation of these diverse use cases demonstrates the reusability and scalability of the graph-based methodology. The overarching aim is to establish the foundations of this evolving approach to knowledge modelling, which next-generation digital Decision Support Systems (DSS) are likely to depend on.

Looking ahead, the research aims to prepare an information layer capable of supporting the next frontier in digital building management: knowledge-based systems integrated with AI. For this reason, the concluding part of the book explores how the methodologies described can pave the way towards AI-integrated knowledge systems. The core idea is that a rich, semantically robust building KG can provide an ideal substrate for emerging AI technologies, enabling natural-language-based modelling and thereby making interaction with building data more intuitive (d'Aquin 2025). The vision is of interactive human-machine systems for the built environment, in which stakeholders, with in-depth domain expertise but low digital skills, can communicate with a building's DSS in natural terms, ask questions, receive answers grounded in a reliable knowledge base, and obtain suggestions or predictions generated by AI (Arazzi *et al.* 2025).

In this perspective, KGs can help AI systems reason with domain-specific rules and avoid propagating errors. It is therefore essential that domain experts – architects, engineers, and heritage conservators – proactively shape the knowledge bases that feed such systems, defining scientifically sound validity criteria, so that increased automation does not come at the expense of rigour and reliability.

The book is organised into three main sections. *Section I* introduces the theoretical foundations of graph thinking and its relevance to building knowledge. *Chapter 1* traces the development of graph-based approaches across different disciplines. In contrast, *Chapter 2* examines their application within the AECO domain, reviewing the state of the art in the use of graph data structures for building-related problems. *Section II* presents the core methodology. *Chapter 3* details the federated-ontology approach, which defines the schema for the KG described throughout the book. *Chapter 4* focuses on implementation and outlines

the software architecture of the graph-building toolkit. *Chapter 5* explores an advanced concept emerging from the methodology – ‘Topological BIM’ – that enables partial automation of knowledge graph construction by deriving semantic structures from building topology. *Section III* contains five application-focused chapters corresponding to the demonstrations outlined above. These chapters illustrate how the methodology can be applied to different facets of building performance and management.

The book concludes with a chapter that summarises the main contributions and discusses how the developed framework can interface with AI. This final chapter reinforces the book’s central message: that embracing *Buildings as Networks* – enriched by ontologies today and by AI tomorrow – can fundamentally enhance human capacity to represent and understand the built heritage domain digitally.

CONCEPTUAL AND ECHNOLOGICAL FOUNDATIONS

1. THE EMERGENCE OF GRAPHS IN SCIENCE AND TECHNOLOGY

Abstract

This chapter explores the conceptual foundations of *graph-based knowledge modelling*, which involves using nodes (entities) and edges (relationships) to represent complex domains. This approach has recently become a cornerstone of digital innovation across multiple sectors.

The chapter begins by introducing the notion of *ontology thinking* for knowledge management and process digitalisation, explaining how formal ontologies structure domain knowledge for both human comprehension and machine reasoning. It then examines the rise of graphs in science and technology as flexible data structures for knowledge representation, highlighting examples from diverse fields, spanning from biology to social science. Subsequently, an overview of graph data science is presented, describing different types of networks and their purposes.

Throughout, graph-based approaches are compared with traditional relational and document-based data models in terms of flexibility, heterogeneity, querying capability, inference, and explainability. Finally, the discussion bridges the topic to the AECO sector, anticipating the main challenges and opportunities associated with the use of graphs in this field, which will be explored in greater depth in the following chapter.

1.1 Ontology thinking for knowledge management and process digitalisation

Digital transformation of complex domains requires explicit representations of knowledge that both humans and machines can interpret. Such human- and machine-readable representations can be established by relying on *ontology thinking*, which refers to the practice of formally defining the concepts, relationships, and rules of a knowledge domain to enable its management and digitalisation.

The term *ontology* has ancient origins. It originally comes from philosophy, where it means the study of ‘being as such’ and of its fundamental categories¹. Implicit in this definition is a logic of decomposing reality and its being into entities and categories, an operation that humans have needed since antiquity to explain the world and the surrounding nature. The term underwent a marked evolution starting in the last century, first within Information Technology (IT) and, more recently, returning to prominence with the progress of Artificial Intelligence (AI). In these fields, the concept was adopted to refer to a structured representation of concepts and their relationships within a domain.

Early knowledge-representation systems (such as semantic networks) laid the groundwork for the emergence of ontologies in IT. Although the word ‘ontology’ was not widely used in computing at the dawn of the IT era, ontologies existed implicitly as vocabularies and structures for expert systems. In 1991, Tom Gruber gave the most widely cited definition: “*an ontology is an explicit specification of a conceptualisation*”.² From those years onwards, ontologies became central to knowledge engineering systems. Reusable ontologies began to be developed for specific domains, like medicine, biology, and engineering. They then gained prominence in the late 1990s-early 2000s with the rise of the Semantic Web, including the *Resource Description Framework* (RDF)³ and the *Web Ontology Language* (OWL)⁴, which, thanks to the efforts of the World Wide Web Consortium (W3C)⁵, became standards for structured data and reasoning on the web. Building on these technological advances, ontologies began to support a wide range of IT applications, such as information retrieval and search engines, data linking and inte-

gration, bioinformatics, and enterprise systems. More recently, ontologies have been coupled with *Knowledge Graphs* (KGs), which represent real-world entities and their relationships as semantic networks.

Thus, an ontology in computer science can be defined as an "*explicit specification of a conceptualisation*": essentially a formal classification of the entities, properties, and relations presumed to exist in a domain. In simpler terms, an ontology provides a shared vocabulary and structure for modelling domain knowledge. By capturing consensus definitions of concepts – for example, in the AECO context, what counts as a ‘building’, a ‘pace’, or a ‘wall’ – ontologies ensure that data and processes across different systems refer to the same underlying ideas. This is extremely important for knowledge management, where the ontology acts as a semantic backbone, enabling the integration of heterogeneous information and preventing misunderstandings arising from inconsistent terminology —among humans, among machines, and between humans and machines.

The intrinsic nature of ontologies enables them to be used not only to represent a domain theoretically but also to support the digitalisation of processes from a technical and technological perspective. This is achieved by making implicit domain expertise explicit and machine-readable. When business or research processes are digitised – whether in healthcare, finance, or construction – the rules and concepts governing those processes must be encoded in software. This is essentially a knowledge-transfer process in which the machine serves as the vehicle for conveying highly specialised domain knowledge to human recipients, who are often not ontology experts but simply users seeking services.

Ontology-driven models facilitate knowledge encoding by providing formal constraints and relationships that software can use for reasoning and validation. For example, an ontology in the field of building operations might specify that an entity definable as `Sensor` is connected to another entity definable as `Space` via a containment relation `HasLocation`. That `Space` may in turn be connected to another entity, for instance a `Building`, via another containment relation, or connected to another `Space` via an adjacency relation, `IsAdjacentTo`. Such specifications not only document knowledge for human stakeholders

but also enable automated inference⁶: if a sensor is in <RoomA> and measures temperature, a query can infer that <RoomA> has a temperature reading, even if not explicitly stored, thanks to the ontological relationships. This ability to derive new facts from existing data, called 'entailment', is a key benefit of ontology-centric knowledge representation. It goes beyond the capabilities of traditional data models by enabling *logical reasoning*, an essential feature for advanced decision-support systems (DSSs) and intelligent automation.

It is for these reasons that the roots of ontology thinking in computer science trace back to early work in AI and knowledge engineering, which recognised that capturing expert knowledge in formal ontologies could significantly improve knowledge sharing and reuse. Today, ontology languages such as OWL allow domain experts to encode rich semantics in a machine-interpretable form. These ontologies, especially when coupled with graph-based data structures, become powerful tools for managing interrelated information, bringing several advantages for knowledge management and digital workflows. These include flexibility to accommodate evolving domain knowledge, support for heterogeneity to integrate different data sources, and logical querying and explainability through semantic meanings and relationships.

Ontology-driven thinking complements, rather than replaces, other data-modelling approaches. While relational databases focus on storing data in structured tables, and document databases store semi-structured records, ontologies focus on the meaning of data and the connections between pieces of information. They encourage knowledge modellers to ask: '*What are the key concepts in this domain and how do they relate?*' This approach naturally leads into *graph thinking* – that is, the use of *nodes* (entities) and *edges* (relationships) to map a domain – being graphs direct, practical ways to represent entities and relationships as defined by ontologies.

In the next section, we delve into how graph-based data structures have risen to prominence for knowledge representation across science and technology.

1.2 The rise of graphs in science as a means for knowledge representation

When people explain ideas or processes, they often implicitly create graphs. In other words, they identify objects or entities and draw connections between them to show how they relate within a system. By representing key entities as nodes and their interconnections as edges (Figure 1), graphs provide a natural and intuitive way to model *complex systems*⁷. Indeed, graph diagrams can concisely capture relationships within a problem, making abstract concepts easier to visualise and communicate, even when the underlying ideas are complex.

Graphs, in the form of networks of nodes connected by edges, have long been used as abstractions in mathematics; however, their adoption as practical data structures for knowledge representation has surged in recent decades. One reason for this rise is the growing recognition that many real-world systems are inherently graph-structured: social relationships, biological pathways, transportation routes, communication links, and more all form networks.

Across the sciences, graph-based thinking has gained prominence as a means of dealing with complexity. In biology, for example, networks are used to represent protein–protein interactions, gene regulatory networks, and food webs, helping researchers uncover key influencers in cellular processes or ecosystems. In social science, social networks map the relationships and interactions among people or organisations; network analysis reveals community structures and influential actors, for instance, in studies of information spread or epidemiology. In transport and logistics, graph models underpin routing and navigation, from airline networks to urban road maps.

Network science has therefore emerged as an interdisciplinary field, leveraging graph theory to study recurring patterns across diverse domains. These approaches are widely applied to predict network behaviour, identify critical nodes or connections, and optimise system performance⁸ (Barabási and Pósfai 2016). A key driver behind the rise of graphs is the explosion of data complexity in the modern era. Traditional data models, such as relational tables, struggle to accommodate the heterogeneity and interconnectivity of big data generated in fields like

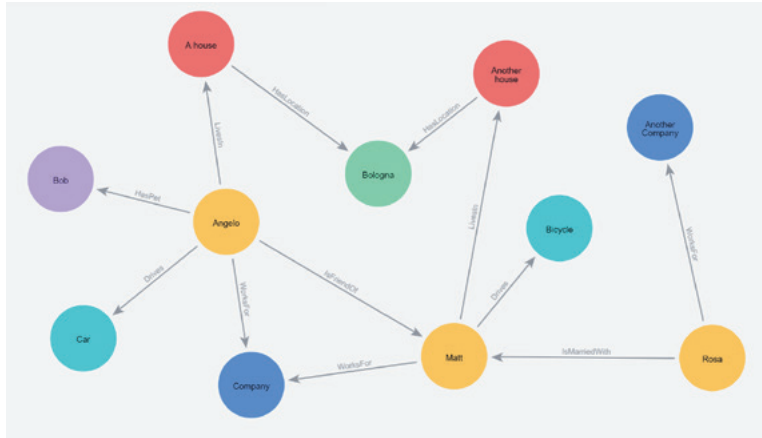
genomics and social media. Graphs, instead, excel at capturing *relationships* as first-class data, rather than flattening connections into foreign keys or array fields. This makes them ideal for knowledge representation where context and linkage are crucial⁹.

The versatility of graphs lies in their abstraction: almost any set of entities with relationships can be mapped to a graph. This universality is coupled with a rich body of algorithms (from graph theory and network science) that can be applied once data is in graph form. For example, an algorithm to find the shortest path between nodes can aid route finding in transportation; the same basic algorithm can also infer molecular pathways or optimise flows in supply chains.

Today, graphs serve both as a means of conceptualising problems and as concrete data structures in software systems. In chemistry and materials science, graph models (such as molecular graphs or crystal graphs) represent atoms and bonds, enabling cheminformatics and the prediction of material properties. In linguistics and natural language processing, graphs appear as parse trees or KGs that connect words and concepts (e.g., WordNet or concept maps in educational technology). Even in physics and engineering, complex systems (power grids, supply chains, ecological networks) are analysed via graph representations to identify vulnerabilities or optimise resilience.

Crucially, graph structures are not only functional for *knowledge visualisation* but also empower *topological analysis*. Modern graph databases and processing frameworks can store massive networks with billions of connections. They enable queries that traverse arbitrary hops (e.g. ‘find how person A is connected to person B through co-authorship and institutional collaborations’), or detect emergent patterns (communities, influencers, anomalies) in ways that relational databases would find inconvenient. As a result, graphs have become foundational in representing, querying, and reasoning over interconnected information in the contemporary data landscape. This has led to a paradigm shift in which graph thinking – viewing data as a network of relationships – is a robust approach for tackling complex, connected problems in science and technology.

Figure 1. Illustrative example of a network represented as a graph.



1.3 Basics of graph data science

1.3.1 Graph data science and its subdomains

As graph representations proliferate, a new interdisciplinary field, Graph Data Science (GDS), has emerged to study and exploit the rich information in network data. GDS can be defined as the collection of theories and techniques for creating, manipulating, analysing, and extracting knowledge from structured data represented as a graph (Raj 2025). It sits at the intersection of classical graph theory, database technology, and modern data analytics, containing many sub-fields (Figure 2).

Graph algorithms, shown on the left of the Venn diagram, are computational procedures that operate on graph structures and process graph data for various purposes, such as solving shortest path problems, detecting communities, and clustering (Wang *et al.* 2022).

Graph visualisation, on the right, consists of methods for visualising structured information as diagrams of abstract graphs and networks (Chen *et al.* 2019).

Graph queries, at the top of the figure, consist of the methods used to retrieve information in graph databases. These databases store data in a graph format. Through various query types (e.g., traversal, pattern matching, and aggregation), users

can search for specific data and retrieve their relationships and properties within these graph structures (Sharmila and Kavitha Devi 2023).

At the intersection between graph algorithms and queries, *graph analytics* refers to the techniques for analysing graph structures and associated data to discover patterns and insights in large datasets. There are many techniques used in graph analytics, such as centrality analysis, community detection, and motif analysis (Ren *et al.* 2023).

Graph data discovery, through queries and visualisation, involves techniques that leverage graph structures and algorithms to understand and explore relationships and connections within large datasets.

Finally, *feature engineering*, which involves integrating graph algorithms and queries, includes methods that leverage relational features in graph datasets to enhance feature extraction. Graph feature engineering involves defining specific graph features, creating feature vectors, and building models using Machine Learning (ML) techniques, such as Graph Neural Networks (GNNs), based on these feature vectors.

1.3.2 Graph structures

From the ‘structural’ point of view, graphs consist of two sets. In mathematical terms, they can be represented as $G = (V, E)$, where the set of nodes is $V = \{v_1, v_2, \dots, v_n\}$, and the set of edges is $E = \{e_1, e_2, \dots, e_m\}$.

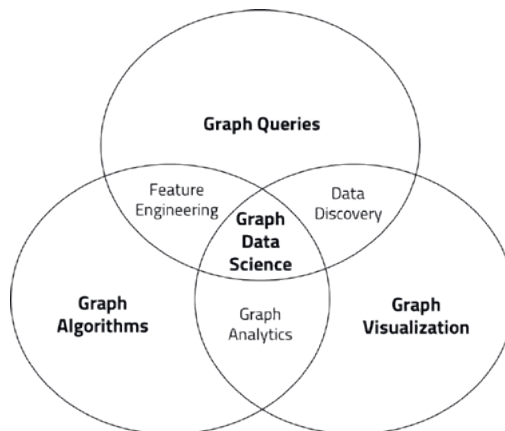


Figure 2. Graph data science sub-fields. Graphic rework from Das and Soylu (2023)

Graphs can be categorised into several fundamental types based on their structural properties, as illustrated in Table 1.

A *null graph* is a graph containing only vertices and no edges, whereas a *simple graph* is an undirected, unweighted graph without loops or multiple edges. *Weighted graphs* are graphs in which numerical values are assigned to edges, often representing distance or connection capacity.

From the perspective of connectivity, a graph may be *connected*, where every node is reachable from any other, or *disconnected*, where such paths do not exist for all pairs of nodes. A *regular graph* is a uniform network where each node has the same degree, that is, the same number of neighbours.

Directionality further differentiates graph structures: a *directed graph* employs ordered pairs of nodes connected by directed edges, while an *undirected graph* treats connections symmetrically, making the order of nodes irrelevant¹⁰.

Beyond the basic structural categories, graphs can also be distinguished by more advanced properties (Table 2).

Heterogeneous graphs incorporate multiple node and edge types, enabling the representation of complex systems in which diverse entities and relationships coexist. *Multidimensional graphs* extend this idea by allowing multiple types of relationships to exist simultaneously between a pair of nodes.

In *signed graphs*, each edge carries a positive or negative value, often used to model supportive or antagonistic relations. *Dynamic graphs* evolve over time as new nodes and edges are continuously added, capturing the behaviour of networks that change in real time, such as social interactions or communication systems.

Hypergraphs generalise the notion of edges by allowing hyperedges that connect more than two nodes simultaneously, thus accommodating group-based relationships. *Marked graphs* distinguish between places (states) and transitions (events), which are linked by directed edges, and are particularly useful for modelling concurrent processes.

Finally, *trees* represent a non-cyclic, connected graph with N nodes and $N-1$ edges, widely employed to organise data hierarchically, for instance, in taxonomies or file system structures.


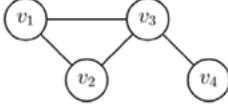
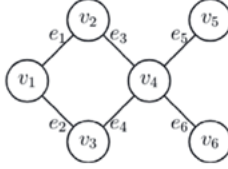
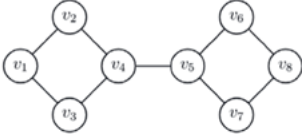
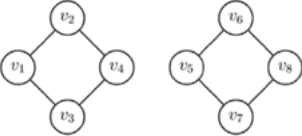
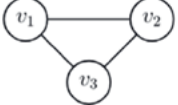
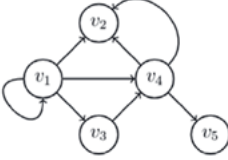
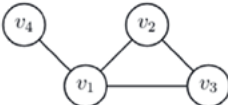
Graph Type	Graphical representation	Properties
Null		A null graph is a graph in which there are no edges between its vertices.
Simple		A simple graph is an unweighted, undirected graph containing no graph loops or multiple edges.
Weighted		A weighted graph is a type of graph in which a numerical weight is given for each edge. So, a weighted graph is a special type of labelled graph in which the labels are numbers.
Connected		A connected graph is a graph in which there is an edge from any node in the graph to any other node.
Disconnected		A disconnected graph is a graph in which there is no edge from any node in the graph to any other node.
Regular		A regular graph is one in which each node has the same number of neighbours; that is, every node has the same degree.
Directed		A directed graph (or digraph) is a set of nodes and a collection of directed edges, each of which connects an ordered pair of nodes.
Non-directed		An undirected graph is a type of graph in which the order of the nodes connected by an edge is unimportant.

Table 1. Fundamental structural types of graphs. Graphic rework from Das and Soylu (2023)

Table 2. Complex graph types. Graphic rework from Das and Soylu (2023)

Graph Type	Graphical representation	Properties
Heterogeneous		A graph that contains different types of nodes and different types of edges.
Multidimensional		A graph where more than one relationship can exist simultaneously between a pair of nodes.
Signed		A graph in which each edge has a positive or negative sign
Dynamic		A graph where new nodes and edges are added to the graph all the time. Such graphs evolve continuously in the process.
Hypergraph		A hypergraph $H = (V, E)$, is a structure consisting of a set of V elements called nodes and of a subset of E of the V elements called hyper-edges. A hyperedge is an edge that joins more than two nodes.
Marked		A marked graph is a Petri net consisting of two different types of nodes: places (states) and transitions (events), along with the directional edges connecting these nodes.
Tree		A tree is a non-cyclic graph, that is, a connected graph with N nodes and $N-1$ edges. It is used to show a hierarchical structure among various data items and organises data into branches that relate to information.

1.3.3 Graph models

A *graph model* provides a clear, intuitive way to represent complex relationships between entities. Like a graph, it uses nodes to represent objects and edges to represent their connections, but it extends this approach by attaching attributes and labels to nodes. *Attributes* describe properties of nodes and edges, such as the area of a room or the width of a window, while *labels* assign names that clarify the structure of the graph and add semantic meaning. By linking graph structures with ontologies, graph models gain semantic depth: graphs provide the network structure for organising data, while ontologies define the meaning, semantics, and logic behind, providing a foundation for computer systems to interpret and process knowledge.

Graph models differ from *relational models*, which are more traditional models for data management (Figure 3). In relational models, data is organised into tabular structures (relations) composed of rows (tuples) and columns (attributes), with relationships expressed through keys and enforced by schema constraints. Queries are commonly executed in Structured Query Language (SQL). This is a domain-specific language used to manage data in relational database management systems (RDBMSs), relying on a set of operations and joins to combine data across tables. By contrast, graph models represent data as networks of nodes and edges, where nodes denote entities and edges denote explicit relationships between them. Queries are expressed in graph-oriented languages such as Cypher or SPARQL¹¹, which support path traversal and pattern matching across networks.

This structural difference has practical consequences: relational models are well-suited for highly structured domains with clearly defined, relatively stable relationships, where-

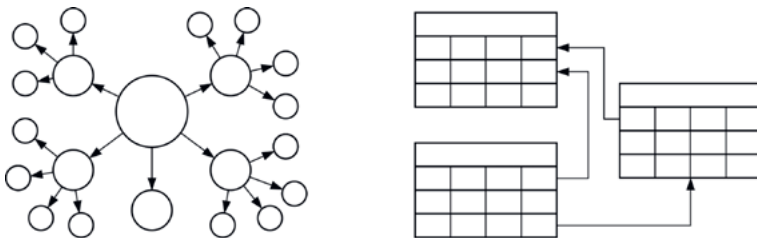


Figure 3. Graph models vs relational models.

as graph models are particularly effective for heterogeneous, semi-structured, or evolving data in which entities are linked by complex, often dynamic relationships. In such contexts, graph models can directly capture interdependencies that are difficult to represent or query efficiently within a relational framework.

Graph models and *graph databases*, while closely related, denote distinct concepts. A graph model is an abstract representation used to analyse the structure and behaviour of networks, often to understand patterns of connectivity or to predict their evolution over time. In contrast, a *graph database* is a database management system specifically designed to persist and manage data structured as graphs. Such systems are optimised for storing nodes, edges, and their properties, and for executing queries that exploit the network structure. Their performance is achieved through graph traversal algorithms and indexing strategies that enable efficient navigation and analysis across highly interconnected data.

Among the different ways of representing knowledge as networks, four models are particularly relevant: *property graphs*, *labelled property graphs*, *RDF graphs*, and *knowledge graphs*. Although interconnected, each of these carries distinct characteristics and typical use cases.

A *property graph* is a graph model in which both nodes and edges can carry additional information expressed as key-value pairs. These properties allow the graph to capture not only the existence of a relationship but also its attributes. For instance, in a collaboration network, a node representing a Person may include properties such as name and affiliation, while an edge representing co-authorship might include a property such as year. Property graphs are valued for their flexibility, since properties can be added without rigid schema constraints, making them effective for integrating heterogeneous data.

Labelled property graphs (LPGs) refine the property graph model by introducing labels for nodes and edges. Labels indicate the type of node or relation (e.g., Person, Organisation, worksFor) and thus provide a lightweight typing mechanism. Like property graphs, LPGs also support properties, combining flexibility with a more structured way of distinguishing entities and relationships. This model is implemented in many graph database systems, such as Neo4j (Neo4j 2025a), and is widely

used for operational applications that require fast traversal and query performance.

RDF is a W3C standard for representing information as directed labelled graphs. RDF expresses data in triples of the form (subject, predicate, object), where each predicate has a defined meaning, often provided by an ontology. Nodes may represent concrete entities or abstract concepts, while edges carry semantic roles. For example, the triple (\langle Giuseppe \rangle , *worksFor*, \langle That-Company \rangle) explicitly states a relationship whose semantics can be shared and extended across systems. *RDF graphs* are typically combined with ontology languages such as OWL, enabling reasoning and inference. This capacity allows systems to derive new facts (e.g., inferring that Giuseppe is an employee if he works for an organisation, which can be defined as an employment relation). RDF thus provides a formal and interoperable basis for semantic data representation.

Finally, a KG can be intended not as a single data model but rather a broader approach that combines graph structures with formal semantics. It represents entities and their interrelations while embedding meaning through ontologies and controlled vocabularies. KGs can be implemented using either RDF or LPG technology, depending on requirements. The choice between them depends on use case: LPGs might be chosen for fast operational queries on richly attributed data, while RDF/OWL graphs might be chosen when interoperability and logical inference are paramount. It is worth noting that the line can blur. One can, for instance, build a KG using a property graph database by imposing a semantic schema on it. Conversely, one can export an RDF graph into a property graph format for conducting specific analyses. KG's defining characteristic lies in the explicit semantic layer, which supports interoperability, reasoning, and integration of heterogeneous data sources¹².

1.3.4 Graph query and analytics

Graphs can be approached from different perspectives. When the focus is on the semantic dimension, graph query languages are most relevant, enabling the retrieval and exploration of knowledge through patterns and ontological structures (Cheng and Sun 2025). Conversely, when attention shifts to the network dimension and the study of interconnected systems (e.g.,

centrality analysis or community detection), the domain of graph analytics becomes central (Bader 2022). In practice, the combination of graph querying and graph analytics provides a powerful synergy: query languages allow precise extraction of relevant subgraphs and data. In contrast, analytics techniques uncover topological patterns and insights.

Graph query languages like SPARQL - for RDF graphs - and Cypher or Graph Query Language (GQL) - for LPGs - allow users to retrieve information by specifying patterns to match in the graph. For instance, a SPARQL query might ask for ‘*all buildings that have at least one sensor that monitors the temperature of a room with an area over 100 m²*’, which leverages the semantic relations defined in an ontology (e.g. building-room hierarchy, sensor placement, and measured parameter).

In general, graph queries are powerful for exploring how entities belonging to different classes are connected. Compared to relational queries, graph queries can be more intuitive for relationship-intensive questions, since they follow network structures. Moreover, graph databases are optimised for traversals and often outperform relational databases for multi-hop queries, which in SQL require numerous join operations.

Graph analytics, on the other hand, encompasses a wide range of algorithms for analysing network topology. These include centrality measures (e.g. *degree centrality*, *betweenness centrality*, and *closeness centrality*), which identify important or influential nodes based on network structure (Figure 4); community detection algorithms (e.g. *modularity clustering*, *label propagation*) that find groups of nodes with dense internal connections (useful for detecting communities in social networks or ‘that’ functionally related clusters in biological networks); pathfinding and traversal algorithms (shortest paths, random walks) for navigation and reachability queries; and subgraph mining for finding recurring patterns or anomalies¹³. Notably, many graph algorithms have no equivalent in purely table-based analysis, or they would be prohibitively expensive to compute without a graph-centric approach.

Given these exceptional capabilities, GDS has expanded in recent years to include ML techniques for graph-structured data (Wu *et al.* 2022). *Graph embedding* techniques learn to represent nodes (or entire subgraphs) as vectors in a continuous space, so

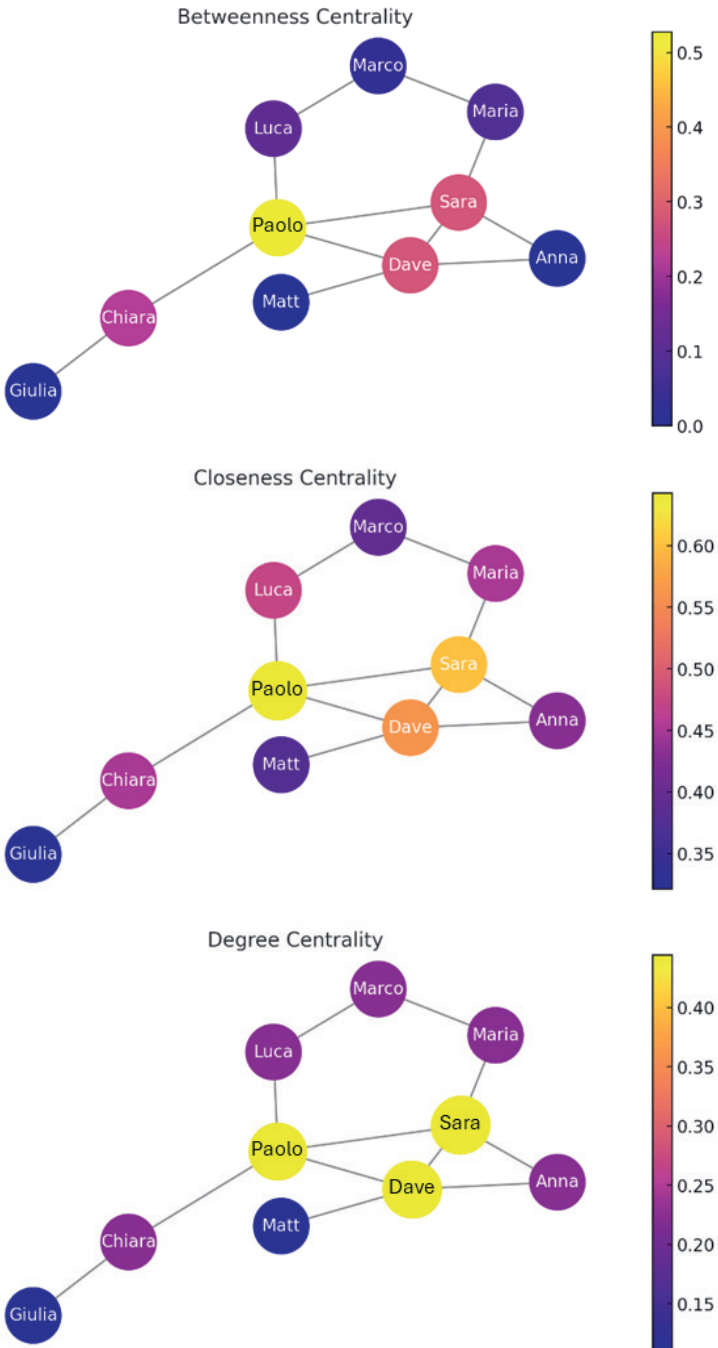


Figure 4. Betweenness, closeness, and degree centrality in a social network.

that geometric relations between vectors reflect network relations in the graph. The purpose of graph embeddings is often to capture latent features of the network (such as community membership or role similarity) in a form that can feed into predictive models. These embeddings enable the use of standard ML algorithms on graph data, such as node classification (predicting a node's label from its embedding) and link prediction (determining whether an edge should exist between two nodes). As part of feature engineering in GDS, analysts may compute embeddings or graph-based features (e.g., the number of connections to a specific node type) to improve predictive accuracy in ML tasks. Techniques such as GNNs have demonstrated impressive capabilities across domains such as protein interaction prediction, social network link recommendations, KG completion, and architectural design (Jabi and Alymani 2020).

1.3.5 Advantages of graph databases

To highlight the distinctive strengths and limitations of graph databases, it is instructive to compare them with more traditional database types.

The first strength of graph databases is their schema flexibility. After a graph database is created, new node or relationship types can be added, or new properties can be added to existing entities, without altering the database schema or migrating to a new one. In contrast, relational databases require a predefined schema of tables and columns; altering this schema (to accommodate a new relationship type, for example) can be unwieldy. In other words, graphs excel at evolving with the domain: if a new relationship or entity emerges, it can be captured as another node/edge without disrupting existing structures. This is crucial in knowledge domains that are not fully understood at the start of a project, or that are rapidly changing, which is common in 'complex' environments.

Second, graph databases easily accommodate *heterogeneity*. The same graph can represent multiple and diverse types of entities: for example, people, organisations, places, events, and documents, all linked in various ways. Because each node's meaning is given by its connections and labels – labels supported potentially by an ontology – mixing different entity types is an intrinsic characteristic of graphs. Relational models, on

the other hand, might require dozens of tables and join tables to achieve the same web of interconnections. Graphs have also demonstrated the ability to serve as ‘information integrators,’ linking structured, unstructured, and semi-structured data within a single network¹⁴.

Third, graph query languages allow *pattern matching* directly on relationships. This can yield more straightforward, intuitive queries for relationship-focused questions. For example, finding a connection between two distant entities is straightforward with a graph traversal query, but in relational databases, it might require recursive self-joins or stored procedures. Moreover, graph databases are optimised for traversals, ensuring that querying deep relationships remains efficient even as data size grows. Relational performance typically degrades as joins become more complex or hierarchical.

Perhaps the most distinctive advantage of semantic graph-based systems is built-in *inference*. By applying ontologies and logic, graph systems can derive new facts from existing ones (as discussed earlier). Neither relational nor document databases have inherent reasoning capabilities; any inferencing must be implemented in application code. In a KG, one can declare rules or class hierarchies, e.g. ‘*all emergency exits are doors*’, and the system can automatically classify all Exit instances as a subclass of Door. This ability to automate reasoning and consistency checking is invaluable in domains where knowledge is complex and can benefit from automated deduction.

Inference also contributes to *explainability*. Beyond formal inference, graphs in general offer more transparent data relationships. If an analysis finds that <ComponentA> is critical to system failure, a graph representation can show exactly which dependencies link the component to the outcome, forming an explanation chain. Relational databases can store the same information, but the connections are implicit in foreign keys and table designs, not as readily visualisable as a network. In contexts like KGs for enterprise data, for instance, users often navigate the graph to understand why a specific recommendation was made or how two pieces of information are related. This mirrors human cognitive reasoning – we often explain things via connections, e.g. ‘*the reason the room is cold is that <Sensor1>, located in that room, reports 15°C and the heating system linked*

to that room is off'. Moreover, this way of reasoning is advantageous not only for human users but also for machines, as KG can be effectively integrated with AI methods, including Large Language Models (LLMs), to support richer and more interactive human-machine interaction¹⁵.

Despite these advantages, some trade-offs remain. Relational databases have benefited from decades of development, resulting in highly optimised systems with robust tooling and practices. Graph databases, by contrast, are comparatively recent. Although they have advanced rapidly in the past decade, managing truly large-scale graphs with billions of nodes and edges often requires distributed graph processing frameworks or specialised graph engines. In comparison, techniques for partitioning and sharding relational databases are long-established and widely adopted.

For this reason, many contemporary knowledge-driven applications adopt a *hybrid architecture*: a graph database captures core relationships and semantics, while a relational or document store manages raw data or high-volume records. This division of labour allows each technology to be employed where it is most effective: graphs for connectivity, integration, and reasoning; relational or document stores for transactional performance and storage of unstructured data.

1.4 Bridging to AECO

The growing prominence of graph thinking in science and technology has not gone unnoticed in the AECO sector. This industry, facing ever-increasing complexity in building information management and lifecycle data, has begun adopting graph-based approaches to model the rich relationships inherent in built assets and their contexts.

In recent years, graph data structures have been employed to digitally represent knowledge about buildings, their components, and related processes throughout the building lifecycle. This includes using graphs to map building elements and their connections (spatial adjacency, functional dependencies) and to integrate diverse data sources such as building information models (BIM), sensor readings, maintenance records, and user

information (Peng *et al.* 2024; Teclaw *et al.* 2024; Wang and Sacks 2025).

One of the earliest applications of graphs in architecture is undoubtedly linked to the spatial analysis of buildings and the representation of systems of spatial connectivity, as in *space syntax* (Mohamed and Van Der Laag Yamu 2024). In this context, graphs provide a powerful means of modelling how rooms and circulation routes are organised and interrelated, offering valuable insights into accessibility, visibility, and functional performance of buildings. When combined with ML techniques, graph-based representations acquire further potential, for example, in the generative design of new floor plans (Nauata *et al.* 2020). Here, neural networks trained on graph encodings of spatial relationships can produce novel configurations of rooms and circulation systems, not simply by imitating existing plans but by learning relational patterns (e.g., adjacency between functional areas, hierarchical organisation of spaces) that characterise effective architectural layouts.

Spatial analysis and generative design do not exhaust the role of graphs in architecture and the AECO sector. An equally notable trend is the increasing use of KG to support digital building management tasks (Ramonell *et al.* 2023).

Like other graph models, these building KGs represent entities (rooms, equipment, people, documents, etc.) as nodes and their interrelations (e.g., *isAdjacentTo*, *feedsInto*, *monitoredBy*) as edges. Beyond that, they embed semantic meaning by building domain ontologies to support inference and queries. The main advantage in this perspective is data integration and interoperability.

Building data traditionally exists in silos – CAD drawings, BIM files, spreadsheets, databases, PDF documents – each capturing a fragment of the whole (Merino *et al.* 2023). Graph-based integration can provide a pathway to link these silos. For instance, a KG can link a BIM model's elements (walls, windows, HVAC units) to real-time IoT sensor data and to maintenance logs, enabling queries that span across these sources (like '*find all rooms on the second floor where temperature is below 19°C and the last window inspection date is over a year ago*'). Early research and industry prototypes have demonstrated graph-based digital twins of buildings, where a graph database serves as a unified

semantic layer integrating static models and dynamic sensor data (Ramonell *et al.* 2023). In such systems, graph queries and analytics help facility managers identify patterns that would be hard to discern from disconnected datasets.

Graph thinking offers a way to reconceptualise building data not as isolated files or tables, but as a knowledge network of building components, stakeholders, and processes. This paves the way for more systemic analysis. The next chapter will delve into this topic in depth, examining how graphs are explicitly applied in the AECO and built heritage domains.

Notes

¹ The word derives from the Greek ὄντος, ὄντος (genitive singular of the present participle of the verb εἶναι, εἶναι, 'to be') and λόγος, λόγος ('discourse'), and therefore literally means 'discourse on being'. It can also derive directly from τὰ ὄντα, that is, 'entities', variously interpreted according to different philosophical positions. (Pianigiani 2022)

² «An ontology is an explicit specification of a conceptualisation. The term I borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what 'exists' is exactly that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is referred to as the universe of discourse. This set of objects, and the formalised relationships among them, is reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions, or other objects) with human-readable text describing what the names are meant to denote, and formal axioms that constrain the interpretation and well-formed use of these terms.» (Gruber 1993)

³ The Resource Description Framework (RDF) provides a foundation for encoding, sharing, and reusing structured metadata, introducing structural rules that ensure semantics can be expressed unambiguously. It offers a mechanism for publishing vocabularies that are simultaneously human-readable and machine-interpretable, thus fostering the reuse and extension of metadata across diverse information communities. (Miller 1998)

⁴ The Web Ontology Language (OWL) is a semantic language designed for creating, publishing, and sharing ontologies on the World Wide Web. Built as an extension of RDF (Resource Description Framework) and derived from the earlier DAML+OIL language, OWL provides a vocabulary for expressing structured knowledge. (Dean and Schreiber 2004)

⁵ The World Wide Web Consortium (W3C) is an international community that develops open web standards, like HTML, to ensure the long-term growth and interoperability of the web.

⁶ Inference refers to the automated reasoning process by which software systems derive new knowledge or establish implicit relationships from explicitly encoded ontological specifications. Specifically, inference enables computational systems to deduce implicit facts from explicitly stated relationships and constraints, apply logical rules systematically to generate new conclusions, and establish transitive connections between entities through chains of defined relations.

⁷ A complex system is «a system is two or more components that combine together to produce from one or more inputs one or more results that could not be obtained from the components individually». (Grieves and Vickers 2017)

⁸ For instance, analysing a transportation network graph can highlight which streets are most central to traffic flow, critical for congestion management, while a graph of scholarly citations can identify influential papers or emerging research communities.

⁹ As one example, the concept of the knowledge graph rose to fame when Goog-

le introduced its Knowledge Graph in 2012, which moved beyond keyword search to connecting facts about real-world entities (people, places, things) in a graph of relationships. Following Google, many technology companies and scientific projects built their own knowledge graphs, making ‘knowledge graph’ nearly synonymous with structured, linked knowledge in the 2010s. (Singhal 2012)

¹⁰ For a further technical description of graph types, it is suggested to see the review article from Das and Soylu, ‘A key review on graph data science: The power of graphs in scientific studies’. (Das and Soylu 2023).

¹¹ Cypher is a declarative query language designed explicitly for graph databases, such as Neo4j. It enables users to express complex queries about nodes and relationships in an intuitive and human-readable manner, utilising a syntax that resembles natural language. SPARQL is the W3C-standard query language for accessing and manipulating data represented in RDF. It is widely used within the Semantic Web to explore, integrate, and connect data from diverse sources.

¹² Well-known examples include the Google Knowledge Graph (Singhal 2012) and Wikidata (Wikimedia Foundation n.d.), which interlink facts from diverse domains in a way that is both human-intelligible and machine-processable.

¹³ GDS encompasses these as graph analytics, techniques that derive insights, such as identifying key influencers or network vulnerabilities. For example, centrality analysis on a citation network can identify seminal papers in a field (highly cited nodes). At the same time, community detection on a product co-purchase graph might reveal distinct customer segments.

¹⁴ For example, a scholarly knowledge graph could link a ‘Research Paper’ (with metadata from a relational DB) to ‘Authors’ (from an HR system), to ‘Topics’ (from a taxonomy), to ‘Funding Grants’ (from a government XML feed). Achieving this in a relational schema alone would be exceedingly complex.

¹⁵ The next frontier involves integrating graph technologies with machine learning – for example, incorporating KGs into LLMs and other AI systems – and advancing tools that enable non-specialists to interact with graph data using natural language or visual interfaces. We will explore this topic in greater detail at the end of the book.

2. GRAPHS IN THE AECO DOMAIN

Abstract

Building on the broader overview above, this chapter introduces, more specifically, how graph data structures, especially knowledge graphs (KGs) and labelled property graphs (LPGs), are being used to manage building information.

The text begins with a brief literature review of graph data structures in AECO, highlighting typical use cases spanning data linking and integration to spatial and topological reasoning. Then it presents some of the most well-established ontologies for building performance management, which underpin building KGs. A discussion about methods and tools for ontology discovery and reuse follows this aspect. Next, the principles and technologies of Linked Data are examined in the context of the built environment.

At the end, the chapter surveys how KGs enable next-generation decision-support systems in building operations. The text highlights implementation patterns, including connector architectures, ontology federation, and hybrid graph-time-series data platforms, and centres on current research and practices that seek to leverage graph-based knowledge modelling in the built environment.

2.1 Uses of graphs in the AECO sector

Graph data structures have gained traction in AECO due to their ability to model complex, interconnected information inherent to buildings and their lifecycle.

From a *data management* perspective, a key driver of the use of such data structures in this field is the need to manage the many relationships among the components that characterise building management: construction components, spatial elements, systems and equipment devices, activities, stakeholders, users, etc (Pauwels *et al.* 2022). From a *technological implementation* perspective, graph-based approaches seem ideal for supporting modern digital tools such as digital twins (DTs), which often require integrating heterogeneous data streams, usually in real-time or pseudo-real-time (Ramonell *et al.* 2023).

Two primary graph paradigms are employed in AECO: LPGs and KGs. *LPGs* are graphs composed of nodes and edges that can carry both semantic labels and attributes, offering a flexible, direct mapping of building elements, their relationships, and their properties. Using *LPGs* can be helpful for applications where properties are tightly coupled to individual components (for instance, the material, cost, and size attributes of a construction component). *KGs*, on the other hand, embed data in a semantic framework where nodes and edges are typed according to an ontology, a semantic schema that enables logical reasoning and inference over the graph¹.

Two typical problems addressed by graphs in AECO include *data integration* and *spatial reasoning*.

On the one hand, graphs excel at connecting diverse data spread among different formats and players into a single network of information. For instance, a building *KG* can link a *BIM* model to *IoT* data streams, maintenance logbooks, or other external data, enabling unified queries across all information. Graph-based representations can help align the schemas of different design tools, simulation engines, building management systems (*BMSs*), and operational databases, thereby improving interoperability and reducing information loss. Such integration tackles the long-standing issue of ‘fragmented data silos’ in AECO, which graphs can mitigate by providing unifying semantic layers to digital systems. A couple of example graphs used for

data integration tasks, coming from the literature, are shown in Figure 5 and Figure 6.

On the other hand, a second key application of graphs in AECO is spatial and topological reasoning. In this context, graph networks are commonly used to represent spaces as nodes, and their adjacency, or connectivity, as edges. This way of representing buildings underpins applications such as indoor navigation and evacuation route analysis systems, which are intrinsically conceived as networks². In these systems, topological graphs can help check building layouts against requirements (for example, ensuring every space is connected to an exit route, or that fire compartments are correctly isolated), offering a powerful complement to geometric modelling. In the context of spatial reasoning, graphs are also used to represent architectural layout datasets (Figure 7 and Figure 8), which, alongside ML techniques, can enable advanced spatial analytics, such as automating the generation of floor plans.

Among the many uses, graph databases have been used to develop decision support systems (DSS) for building operations, such as DTs. On such platforms, graph databases often serve as the backbone that links static building data (e.g., from BIM or equipment inventories) with dynamic data (e.g., from sensors or other data streams) within a unified model. This semantic integration can allow DT users to query knowledge about built assets spanning multiple domains of information; for example, *'find all occupied rooms that lack adequate ventilation'*.

But applications are not limited only to spatial reasoning and data integration. Given the potential of graphs, the AECO research community has been exploring methods to generate graphs from raw or legacy data to speed up and automate the knowledge modelling process. Examples include parsing floor-plan images, converting laser-scanning point clouds into graphs, converting BIM models into graphs, and automating HVAC topology modelling for building performance simulations (Wang *et al.* 2024). Additional developments over the past few years have led to the emergence of Large Language Models (LLMs) intersecting with graph techniques (Pan *et al.* 2023) such as ChatGPT and GPT4, are making new waves in the field of natural language processing and artificial intelligence, due to their emergent ability and generalizability. However, LLMs are black-

box models, which often fall short of capturing and accessing factual knowledge. In contrast, Knowledge Graphs (KGs). Graphs and their structured knowledge bases indeed proved extremely important for improving the contextual grounding of LLM outputs, enabling LLMs to serve as natural language interfaces for querying graph data. This integration, known as Retrieval-Augmented Generation (RAG), is an emerging area of research also in the construction industry (Uhm *et al.* 2025).

Other emerging patterns in the field, especially in DT development, include blending graphs with time-series data within hybrid data architectures capable of handling both semantics and big data storage (Chamari *et al.* 2022). In such applications, a core KG provides ontology-backed semantics for static and structural information; meanwhile, high-frequency data (e.g., from IoT sensors) is stored in scalable time-series platforms or document databases, with semantic links between objects and measurements maintained in the KG. Examples include energy metering and monitoring systems that link smart meter time-series to building entities through a semantic layer (Chamari *et al.* 2022), indoor air quality (IAQ) monitoring platforms that connect sensor data streams to spatial and equipment models within a DT (Benfer and Müller 2024), and fault detection and diagnostics (FDD) frameworks that exploit hybrid architectures, combining a KG for semantic context and time-series analytics for anomaly detection (Du and Luo 2023).

Despite the several advantages, challenges remain in terms of standardising graph modelling practices across the industry and ensuring these solutions scale with big data, issues that ongoing research and community efforts continue to address.

Figure 5. Example KG structure for heritage building conservation, illustrating the interconnectedness of components, environmental conditions, materials, and digital twin data sources. Source: (Hosamo and Mazzetto 2024)

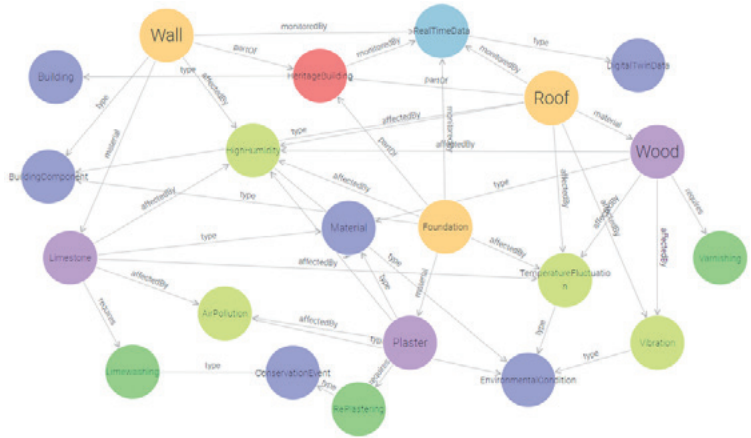
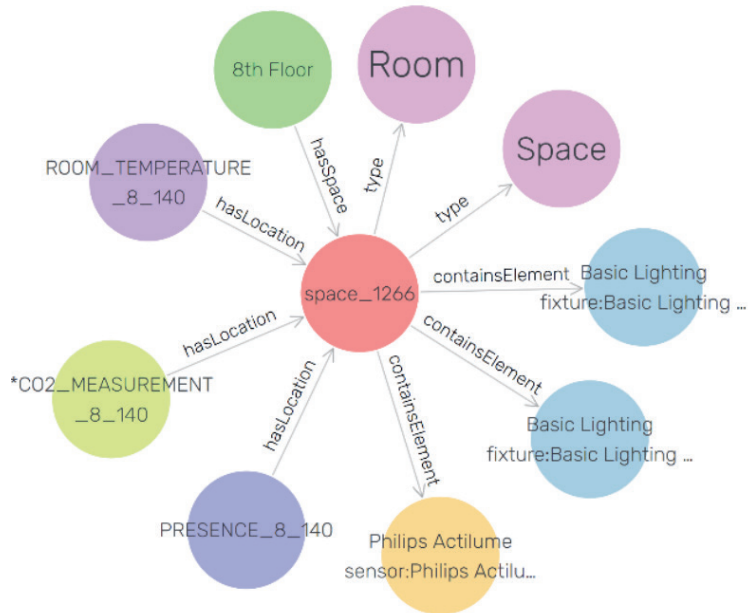


Figure 6. Graph representation of a space in a graph database integrating BIM, BMS and IoT data on the Web. Source: (Chamari et al. 2022)



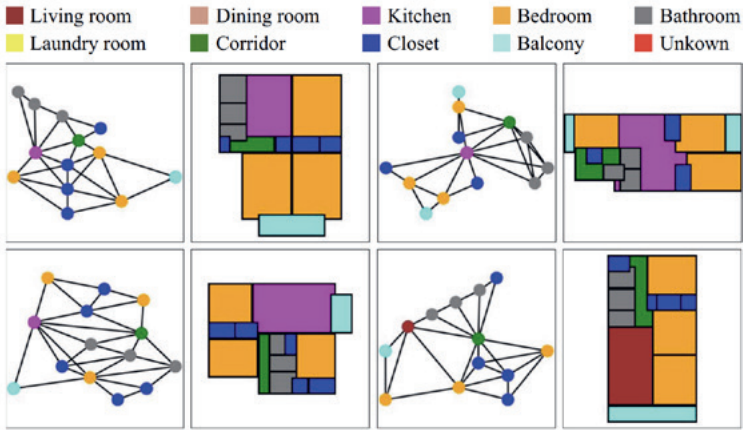


Figure 7. Sample bubble diagrams and house layouts in the HouseGAN dataset. Source: (Nauata et al. 2020).



Figure 8. Sample of the Modified Swiss Dwellings 01 JSON, a JSON derivation of the Modified Swiss Dwellings Dataset for TopologicPy. Source: (Jabi 2025)

2.2 Ontologies in AECO

As discussed in the previous chapter, the concept of a graph is closely intertwined with that of *ontology*. Ontologies provide the semantic schemas that underpin KGs in the building domain, ensuring that entities, attributes, and relationships are formally defined and consistently interpretable. For this reason, research on ontologies has advanced in parallel with research on graph technologies, and in the AECO field, a wide range of ontologies has been developed to support the creation and use of KGs.

Lygerakis et al. conducted a systematic review and identified the most significant ontologies for the building domain, focusing on performance issues – the main scope of this book – and categorising them by lifecycle phase: design or operation (Lygerakis et al. 2022).

Among the ontologies for the design phase (Table 3), a range of complementary ontologies and schemas allows for the modelling and exchange of building information.

The *Industry Foundation Classes* (IFC) are the international standard for BIM, developed and maintained by buildingSMART International. (buildingSMART International 2025) They provide a comprehensive, object-oriented schema for sharing data of the built asset industry. *IfcOWL* provides a Web Ontology Language (OWL) representation of the IFC schema. (Pauwels and Terkaj 2019) Recognising the practical overhead of the complete IFC vocabulary for some tasks, *simpleBIM* offers a streamlined subset of ifcOWL. (Pauwels and Roxin 2016)

For performance analysis, *Green Building XML* (gbXML) provides a schema for data transfer between BIM and Building Energy Models (BEMs). (Green Building XML (gbXML) Schema, Inc. n.d.) Additional ontologies for building simulation include the *SimModel Ontology* (Pauwels et al. 2014) and *EnergyADE*, both of which focus on exchanging simulation inputs and outputs in more semantically explicit forms. Finally, *Tubes* targets the description of building service systems at a high level, helping to capture HVAC and related MEP (Mechanical, Electrical, and Plumbing) logic beyond geometry (RWTH-E3D n.d.).

These ontologies and data schemas only form a small subset of the ontologies for building design.

From the perspective of building operations (Table 4), the number of ontologies explicitly developed for digital building management is substantial.

Some ontologies target the representation of sensing and actuation in smart environments, including *Semantic Sensor Network (SSN)* (Compton *et al.* 2012) and *Sensors, Observations, Samples, and Actuators (SOSA)* (Janowicz *et al.* 2018), *Web Thing Model (WoT)* (EVERYTHING 2017), and the *oneM2M Base Ontology* (oneM2M 2019), which provide formal means to describe sensor networks, virtualised objects, and syntactic interoperability for IoT frameworks.

Others focus more explicitly on energy-aware and intelligent control systems, including *ThinkHome* (Reinisch *et al.* 2010), *Smart Energy Aware Systems* (Kukkonen *et al.* 2022), and the *Ontology of Smart Building (SBOnto)* (Žáček and Janošek 2017), while domain-specific frameworks such as *BOnSAI* (Stavropoulos *et al.* 2012) and *SAREF* (Lefrançois *et al.* 2024) support ambient intelligence and device-level interoperability.

Project Haystack (Community 2025), the *Haystack Tagging Ontology* (Charpenay 2020), and the *Brick Schema* (Balaji *et al.* 2018) have emerged as widely adopted tag-based or ontology-driven standards for describing building equipment, points, and control logic. More recent efforts, such as the *Google Digital Building Ontology (DBO)* (Berkoben *et al.* 2020) and *RealEstate-Core (REC)* (Hammar *et al.* 2019), aim to provide higher-level frameworks for asset management, usage optimisation, and integration with enterprise systems.

Beyond the building scale, ontologies such as *KM4City* (Bellini *et al.* 2018), the *Facility Smart Grid Information Model, EM-KPI Ontology* (Li *et al.* 2019), and *RESPOND* (Esnaola-Gonzalez and Díez 2020) extend these approaches to encompass energy management, grid interaction, and urban mobility.

As the number of ontologies for buildings continues to grow, identifying, selecting, and reusing the most appropriate becomes increasingly challenging³. To mitigate this issue, several initiatives have emerged to encourage ontology discovery and reuse, limiting the development of new ontologies to cases where it is strictly necessary. Among these, the *Built Environment Ontology Lookup Service (BE-OLS)* is an initiative promoted by the EC3 M&S Committee (Arghavan Akbarieh *et al.* 2025).

BE-OLS provides a centralised repository and registry of ontologies relevant to the built environment (Figure 9). Inspired by general ontology libraries such as Linked Open Vocabularies (LOV), it is specifically tailored to the needs of the AEC and Facility Management (FM) community. It aims to support researchers and practitioners in discovering existing ontologies, evaluating their use, and understanding how they may be connected or integrated, rather than duplicating efforts by creating yet more ontologies. BE-OLS aggregates metadata about several AECO ontologies (including scope, publisher, and applications) and offers search functionality by keyword or domain (e.g. ‘HVAC’, ‘geometry’, ‘heritage’). A further feature is ontology evaluation, based on three criteria (i.e., accessibility, connectivity, and reuse & documentation), which guides users to ontologies that are actively maintained or widely adopted.

Table 3. Ontologies for building design reviewed by Lygerakis et al. (Lygerakis et al. 2022)

Name	Scope/Description	Year
Industry Foundation Classes (IFC)	Gives spatial and other properties to every building entity	2013
ifcOWL	Descriptive OWL representation of IFC schema	2016
simpleBIM	Simplified version of ifcOWL	2017
Green Building XML (gbXML)	Information exchange between BIM and Energy Models	2000
Tubes	High-level description of building service systems	2020
SimModel Ontology	Exchange of energy simulation data	2014
EnergyADE	Exchange of energy simulation data	2014

Name	Scope/Description	Year
Semantic Sensor Network/Sensor, Observation, Sample, and Actuator (SSN/SOSA)	Focuses on sensors in buildings	2011
Web Thing Model (WoT)	Model to describe the virtual counterpart of physical objects in the Web of Things	2015
oneM2M BaseOntology's	Provide syntactic and semantic interoperability between oneM2M and external systems	2016
One Data Model (OneDM)	Model to support a common language for the Internet of Things	2018
Smart Energy Aware Systems		2016
ThinkHome	Ontology that includes concepts needed to realise energy efficient and intelligent control mechanisms	2011
Building Ontology for Ambient Intelligence (BOnSAI)	A smart building ontology for ambient intelligence	2012
DogOnt	Model for all devices being part of IoT inside a smart environment	2008
Ontology of Smart Building (SBOnto)	Smart Building Ontology	2017
Smart Applications REFerence (SAREF)	Matches existing assets in the smart applications domain	2014
Project Haystack 3	Hierarchical representation of building entities and concepts utilising tag sets	2014
BASont	Building Automation and Monitoring	2012
Project Haystack 4	Hierarchical representation of building entities and concepts utilising tag sets	2019

Table 4. Ontologies for building operation reviewed by Lygerakis et al. (Lygerakis et al. 2022)

Name	Scope/Description	Year
Haystack Tagging Ontology (HTO)	Streamlining Data from IoT based on Project Haystack	2016
Brick Schema	Metadata and data points from building advancement and needs based on end-use applications	2016
Google Digital Building Ontology	Represent structured information about buildings and building-installed equipment	2020
Semantic BMS ontology (SBMS)	BAS-protocol-independent model of intelligent building systems	2016
CTRLont	Model of Control Logic in Building Automation Systems	2017
Green Button	Building Automation and Monitoring	2011
RealEstateCore (REC)	Usage analysis, optimisation, and presence analysis of a building structure	2017
Building Topology Ontology (BOT)	Representation of physical and conceptual objects of a building and the connections between them	2019
Building Automation and Control Systems (BACS)	Supports the modelling control behaviour in a BAS, physical devices of BAS and their location in the building and connection to technical equipment and appliances	2017
Knowledge Model for City (KM4City)	Representation model for the city and mobility	2014
EM-KPI Ontology	Enhance energy management at district and building levels	2017
Facility Smart Grid Information Model	An abstract information model of what the Smart Grid looks like from the perspective of a facility	2014
RESPOND	Manage real-time optimal energy dispatching, considering all energy assets on-site	2020

Built Environment Ontology Lookup Service Home Knowledge Hub About Us Contact

Search Ontologies ...

BEM-Reno Ontology
BEM-RENO

Primary Domain: Energy
Secondary Domain: N/A
FAIR Score: 15
2020

[See Details](#) [See Ontology](#)

BIMERR Key Performance Indicator ontology
KPI

Primary Domain: Energy
Secondary Domain: N/A
FAIR Score: N/A
2020

[See Details](#) [See Ontology](#)

Building Performance Ontology
BICP

Primary Domain: Energy
Secondary Domain: IoT Sensors/Actuators
FAIR Score: N/A
2021

[See Details](#) [See Ontology](#)

Digital Construction - Energy
DICES

Primary Domain: Energy
Secondary Domain: N/A
FAIR Score: N/A
2019

[See Details](#) [See Ontology](#)

Energy and Resource Information
TH-ENERGY

Primary Domain: Energy
Secondary Domain: N/A
FAIR Score: N/A
2013

[See Details](#) [See Ontology](#)

Energy Efficiency Prediction Semantic Assistant ontology
EPEPSA

Primary Domain: Energy
Secondary Domain: N/A
FAIR Score: N/A
2021

[See Details](#) [See Ontology](#)

Figure 9. Screenshot from the Built Environment Ontology Lookup Service website (September 2025). (EC3 M&S Committee 2025)

2.3 Linked Data for the built environment

2.3.1 Core principles

When discussing graphs and ontologies in support of digitalisation processes, it is essential to address the notion of *Linked Data* (LD), a set of principles and technologies introduced by Tim Berners-Lee (Bizer *et al.* 2009) and developed by the W3C for publishing and interlinking structured data on the web.

LD provides a framework for representing information in graph-based data models, ensuring that entities and their relationships can be consistently interpreted across heterogeneous systems. Its principles are widely regarded as the foundation of the semantic web and are increasingly relevant to digitalisation in the AECO sector.

The *core principles* of LD are:

- Use *Uniform Resource Identifiers*⁴ (URIs) to uniquely name entities so they can be referenced unambiguously across multiple systems;
- Make these URIs dereferenceable using HTTP, enabling humans and machines to look them up and retrieve valuable information;
- Provide information in standardised formats (notably RDF and related serialisations) so that data is machine-readable and semantically interpretable;
- Include links to other URIs to enable discovery of related resources and to interconnect datasets.

Together, these principles ensure that data elements are not isolated records but form a web of data.

2.3.2 Graph serialisation

In the context of LD, *graph serialisation* refers to the process of writing graph data to a file format so it can be stored, exchanged, and processed by machines.

Two popular serialisation formats for graph data are Turtle and JSON-LD. *Turtle* (*Terse RDF Triple Language*) is a plain-text, human-readable format for RDF. It represents triples in a compact form using namespace prefixes. It is commonly favoured by ontology engineers and in documentation because of its

readability and ease of editing⁵. *JSON-LD (JavaScript Object Notation for Linked Data)* is a JSON-based format that allows RDF data to be serialised as JSON documents. JSON-LD is especially useful for linking building data with other web data sources or for retrieving data from web services. It is more common for developers who work with JSON, web APIs (Application Programming Interfaces), and web applications.

The following snippets illustrate how an example Brick graph – containing one storey, one space, and one sensor – can be represented both in Turtle (Figure 10) and JSON-LD (Figure 11).

Two sections characterise both formats. The first contains the context data, which refers to the ontologies used in the graph, mapping JSON keys to RDF IRIs and datatypes, ensuring that the data follows the semantics defined by the ontology, `@prefix` in Turtle and `@context` in JSON-LD. The second section contains the actual data, where objects are mapped along with their IDs, semantic classes, and relationships with other elements (e.g., `<Temp_Sensor_101> hasLocation <Space_101>`).

Both Turtle and JSON-LD allow the same underlying graph to be represented in different ways. The choice may depend on the use case. The choice between the two depends on the specific needs. JSON-LD is ideal for web applications and data exchanges that use JSON. In contrast, Turtle excels in readability and expressiveness, making it better suited for more complex datasets or cases where greater clarity is required.

Figure 10 . Serialisation of an example graph through Turtle.

```

@prefix brick: <https://brickschema.org/schema/Brick#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex: <https://example.org/building#> .

ex:Storey_1 a brick:Storey ;

ex:Space_101 a brick:Space ;
    brick:hasLocation ex:Storey_1 .

ex:Temp_Sensor_101 a brick:Temperature_Sensor ;
    brick:hasLocation ex:Space_101 .

```

Figure 11 . Serialisation of an example graph through JSON-LD.

```

{
  "@context": {
    "brick": "https://brickschema.org/schema/Brick#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "ex": "https://example.org/building#"
  },
  "@graph": [
    {
      "@id": "ex:Storey_1",
      "@type": "brick:Storey",
    },
    {
      "@id": "ex:Space_101",
      "@type": "brick:Space",
      "brick:hasLocation": { "@id": "ex:Storey_1" }
    },
    {
      "@id": "ex:Temp_Sensor_101",
      "@type": "brick:Temperature_Sensor",
      "brick:hasLocation": { "@id": "ex:Space_101" }
    }
  ]
}

```

2.3.3 Query languages

Once building data is in a graph form, query languages are needed to extract it.

SPARQL is the W3C-standardised query language and protocol for retrieving and manipulating data stored in RDF format. It was initially published by W3C as a recommendation in January 2008 (SPARQL 1.0) and later extended to SPARQL 1.1 in March 2013. SPARQL enables users to write queries that operate over RDF's triple-structure (subject-predicate-object). It supports triple patterns, conjunctions, disjunctions, and optional patterns, enabling expressive graph pattern matching and traversal. Unlike conventional relational databases, where the schema is separate, in RDF, the schema (ontology) can be intrinsic to the data. SPARQL can execute operations analogous to joins, sorting and aggregation across graph-structured data, as well as path-traversal queries (including transitive relationships).

SPARQL supports several query forms, each suited to different purposes:

- *SELECT*: extracts raw values in tabular form.
- *CONSTRUCT*: produces an RDF graph as output, enabling transformation of query results into RDF.
- *ASK*: returns a Boolean value (true/false) answering whether the query pattern matches.
- *DESCRIBE*: returns an RDF graph about the resources matched by the query; the actual graph is left to the endpoint to define.

SPARQL 1.1 further introduced additional forms for updating RDF data (INSERT, DELETE) and more advanced features.

Because SPARQL is designed for the semantic web and LD paradigms, it plays a key role in querying and integrating heterogeneous datasets across domains without requiring a rigid, centralised schema. It enables flexible data modelling via ontologies and supports federation across multiple endpoints and datasets (federated queries). A simple SELECT query is provided in Figure 12. In addition to SPARQL, other query languages can be used for extracting graph data. For LPG databases like Neo4j⁶, the primary query language is often Cypher, though alternatives such as Gremlin and GQL (Graph Query Language) are available.

Cypher is Neo4j's declarative graph query language and is compliant with the GQL standard. It provides syntax that is intuitive and visually oriented, using ASCII-art style patterns (e.g., (:node)-[:REL]->(:other)) to represent graph structures. *Nodes* are represented with parentheses, e.g., (n:Label). *Relationships* are represented with square brackets and arrows, e.g., (a)-[:RELTYPE]->(b) for a directed relationship. *Properties* are key-value pairs assigned to nodes or relationships, e.g., {name:'Sally', age:30}.

Cypher uses a graph pattern syntax to find subgraphs within the larger graph. A typical Cypher query might look like: *MATCH (r:Room)-[:CONTAINS]->(s:Sensor) WHERE s.value > 100 RETURN r.name, s.id*. This would return rooms *r* and sensors *s* where the sensor's value is over 100. Cypher tends to be intuitive for those with a lower programming background. In AECO contexts, Cypher is also frequently used for tasks such as pathfinding, centrality analysis, and simple graph operations. An example Cypher query is shown in Figure 13.

Figure 12. Example SPARQL query that retrieves all temperature sensors together with the spaces and storeys in which they are located.

```
PREFIX brick: <https://brickschema.org/schema/Brick#>
PREFIX ex: <https://example.org/building#>

SELECT ?sensor ?space ?storey
WHERE {
  ?sensor a brick:Temperature_Sensor ;
          brick:hasLocation ?space .
  ?space brick:hasLocation ?storey .
}
```

Figure 13. Example Cypher query that retrieves all temperature sensors together with the spaces and storeys in which they are located.

```
MATCH (sensor:Temperature_Sensor)-[:LOCATED_IN]->(space:Space)-
[:LOCATED_IN]->(storey:Storey)
RETURN sensor, space, storey;
}
```

2.3.4 Validation, constraints, and data quality

In a KG representing a building, data validation is crucial to ensure the consistency and reliability of the information. Unlike a traditional relational database, where a schema enforces data types and relationships, graph data can be more flexible, a strength but also a risk if incorrect data slips in.

To respond to the need for *data validation* in graphs, the Semantic Web community has developed shape languages such as SHACL (Shapes Constraint Language) and ShEx (Shape Expressions). The former was designed to enhance the semantic and technical interoperability layers of ontologies expressed as RDF graphs; the latter was a proposed high-level language for RDF validation. These shapes can be used to validate a given KG and catch errors⁷. In the AECO domain, where data comes from many sources, using such constraint checks is a good practice to maintain a high level of trust in the KG.

Identifiers are another critical aspect in graphs, as already seen for Turtle and JSON-LD. To avoid any error, each real-world entity must be unambiguously identified in the graph. In RDF, this means assigning a URI to every entity. Other projects use IRI patterns or tagging schemes. There are also standard identifier schemas, such as GUIDs from BIM models. The important thing is that identifiers are unique, since stable identifiers help merge data from different sources. Good identifier management also facilitates versioning: if a building undergoes renovation and rooms are renumbered, the KG maintainers might choose to mint new URIs for new rooms but keep old ones for historical data, linking them via a ‘wasPreviousVersionOf’ relation or similar.

Data versioning in graphs is essential because building data is rarely static. This is critical for instance data but also for ontologies, since ontologies themselves can change or be updated over time⁸. In some cases, for instance data, capturing snapshots over time can be needed. Some projects adopt a temporal triple approach or use named graphs to denote different states. More simply, maintaining a clear process for updating the KG when source data changes ensures that users of the KG know how current it is and can trace changes.

Other essential data for scaling up KGs are *provenance* metadata. In a federated building KG, where data likely comes from

BIM files, sensor systems, user inputs, etc, using standards or simpler annotations can allow for tagging every triple or node with its source or trust level. Provenance tracking is especially relevant when the KG is used for decision-making, when users need to trust the data.

Together with a robust ontological framework underlying the KG, these practices ensure that, as the KG grows in scope or over time, it remains a reliable resource for decision support.

2.4 Knowledge graph applications for digital building management

2.4.1 Digital twins for the built environment

The value of graph-based modelling in the AECO sector is increasingly evident through the development of applications that support decision-making across the entire building life-cycle. Among these, DTs – virtual replicas that integrate static information with dynamic, real-time data (Grieves and Vickers 2017) – have recently gained prominence in research and now represent a leading example of an emerging paradigm that draws significant benefit from KGs.

In recent years, the DT paradigm has emerged as one of the most promising drivers of the digital transformation across multiple industries. Initially conceived in the manufacturing sector (Grieves and Vickers 2017; Tao *et al.* 2019) (Figure 14), the concept has since been extended to diverse domains, including aerospace, automotive, healthcare, agriculture, environmental monitoring, emergency management, and construction. (Boje *et al.* 2020) Across these fields, the value of DTs is increasingly recognised in their capacity to enhance visualisation, monitoring, and optimisation of products and processes (Gerber *et al.* 2019).

Synthesising various definitions from the literature⁹, a DT may be described as an information environment capable of abstracting, structuring, processing, and visualising data relevant to an object, process, environment, or system that exists – or is intended to exist – in the physical world. Within the AECO sector, DTs are expected to provide the means to describe, inspect,

monitor, maintain, and manage built assets across their entire lifecycle by integrating static data (e.g. BIM–GIS models) with dynamic or real-time data (e.g. IoT devices, sensor networks, or user input). Looking forward, the convergence of DTs with AI is anticipated to enhance their capabilities further, enabling them to reason, learn, optimise, predict, support decision-making, and, ultimately, autonomously adapt or influence their physical counterparts.

Although a variety of definitions exist¹⁰, there is a broad consensus in the literature regarding a set of fundamental requirements that a DT should fulfil. A widely accepted premise is that information must flow from the physical asset to its digital counterpart, which in turn necessitates real-time or near-real-time updates to ensure that the virtual representation accurately reflects the current state of the physical system. Another commonly acknowledged expectation is that a DT should actively support the operation and management of its physical counterpart, for example, by providing predictive insights to inform control strategies, maintenance planning, and component analysis.

To formalise these requirements, Ganguli and Adhikari (Ganguli and Adhikari 2020) identified four essential aspects of DTs:

1. modelling and simulation;
2. data fusion (where the use of KGs can be essential);
3. interaction and collaboration;
4. services.

Complementing this perspective, Boje et al. (Boje *et al.* 2020) proposed a more detailed framework for the built environment, defining nine technical requirements – connectivity, interoperability, extensibility, modularity, reusability, maintainability, scalability, security, and controllability – and fourteen functional abilities, including sensing, monitoring, actuation, integration, storage, sharing, linking, emulation, diagnosis, visualisation, simulation, prediction, optimisation, and reaction.

The survey of DT initiatives across different contexts highlights the diversity of scales, domains, and applications at which such technologies are being deployed.

At the city scale, projects such as *Virtual Singapore* (National

Research Foundation Singapore n.d.), the *Helsinki* (Ruohomaki *et al.* 2018) and *Zurich City Digital Twins* (Schrotter and Hürzeler 2020) and the *Newcastle and Gateshead Flood-PREPARED Project* (Northumbria University 2019) demonstrate the use of DTs for urban planning, infrastructure management, climate and energy simulation, and risk assessment. At the district level, examples such as the *Port of Antwerp* (Port of Antwerp Bruges n.d.), *Queen Elizabeth Olympic Park* (Dawkins *et al.* 2018), and *Trent Basin* projects focus on energy monitoring, logistics, and user engagement, reflecting a growing focus on neighbourhood-scale optimisation and citizen participation. Moving to the building scale, initiatives such as the Milan Cathedral, West Cambridge Campus (Merino *et al.* 2023), and the Stafford Railway Bridge demonstrate how DTs can support structural monitoring, cultural heritage conservation, and facility management. Finally, at the system scale, research prototypes such as those by Xie *et al.* (Lu *et al.* 2020), Matsuda and Ooka (Matsuda and Ooka 2020), and Tran *et al.* (Tran *et al.* 2021) illustrate the use of DTs for equipment-level optimisation, energy performance analysis, and quality control in prefabrication, respectively.

Figure 14. DT conceptual model proposed by Grieves and Vickers (Source: (Grieves and Vickers 2017)). Graphic elaboration from the author



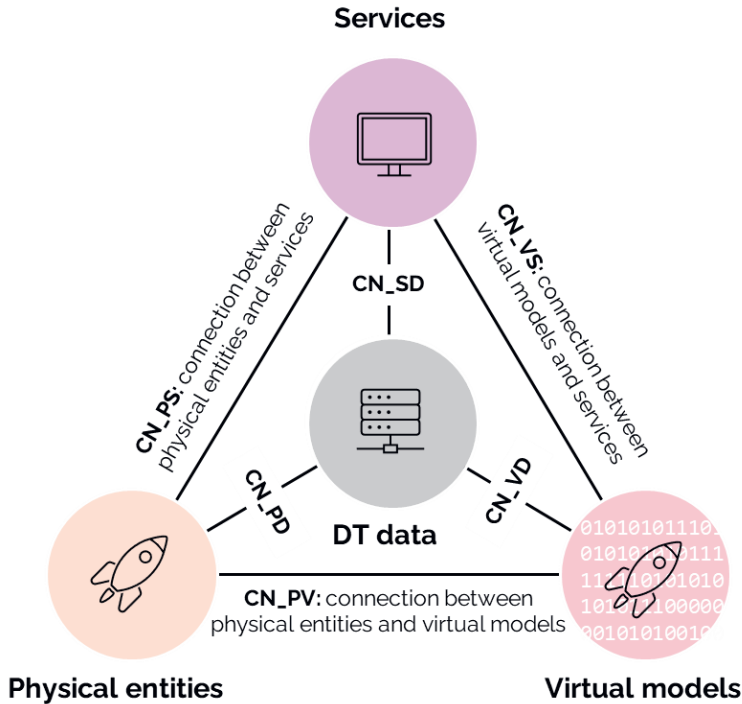


Figure 15. DT conceptual model proposed by (Tao et al. 2019), on the bottom. Graphic elaboration from the author.

2.4.2 Graph-based digital twins

As described by Lu (Lu *et al.* 2022), the system architecture of DTs is often organised as a multi-layered framework composed of several interconnected layers, each with a distinct function in managing the building data lifecycle.

Six consecutive levels can typically be distinguished (Mas-safra 2024) (Figure 16). The *data acquisition layer* (1) captures structured inputs (e.g. BIM and BPS models, SaaS data) alongside unstructured sources (e.g. point clouds, documents, and sensor data). These are then transferred through the *data transmission layer* (2), which manages the heterogeneous requirements of models, documents, and real-time IoT data, with blockchain and open-system APIs ensuring reliability, traceability, and interoperability. Collected information is stored in the *data storage layer* (3), which can rely on data warehouses for structured data and data lakes for raw, unstructured data, with emerging lakehouse paradigms combining both. From there,

the *data extraction and processing layer* (4) employs ETL (Extract, Transformation, Load) tools, APIs, and algorithms to run calculations. The *data integration layer* (5), often supported by LG and LD approaches, then links all data stored in the data storage layer and calculated in the data extraction and processing layer. Finally, the *data visualisation layer* (6) operationalises the architecture by providing front-end services, dashboards, and applications tailored to user needs.

Of particular relevance for this work is the data integration layer, which constitutes the core of this system architecture. At this level, KGs and semantic technologies are employed to interlink heterogeneous datasets and represent Buildings as Networks, thereby enabling structured information management.

This is among the most promising domains for graph database applications. In the context of DTs, graph-based integration establishes a unified data environment in which all sources are connected through relationships. For instance, a digital twin KG may link a Room object (modelled in BIM via ifcOWL/BOT) to a CO₂ Sensor (described with Brick), to the corresponding HVAC Terminal (via Brick), and to the MaintenanceRecord documenting the last filter replacement. Such representation supports cross-domain queries. For example: *'Are rooms with high CO₂ also those with overdue HVAC maintenance?'* That would be far more difficult to answer if the data remained siloed.

A key pattern in graph-based DT platforms is the use of *connector architectures* within the data extraction and processing layer. Connectors are modular scripts or services that translate native data (such as an IFC file, a CSV of sensor readings, or an API from a maintenance system) into entries in the KG via ETL processes. Each connector knows how to map its source's data schema to the KG's ontology.

For instance, an IFC connector might iterate through spaces and equipment in a BIM model and create corresponding BOT and Brick instances in the KG. A sensor connector might pull the latest readings from a time-series database and attach those values to the corresponding Brick *Point* nodes in the KG. By having separate connectors, the system can be flexible and extensible, and new data sources can be added by writing a new connector without disrupting the overall architecture.

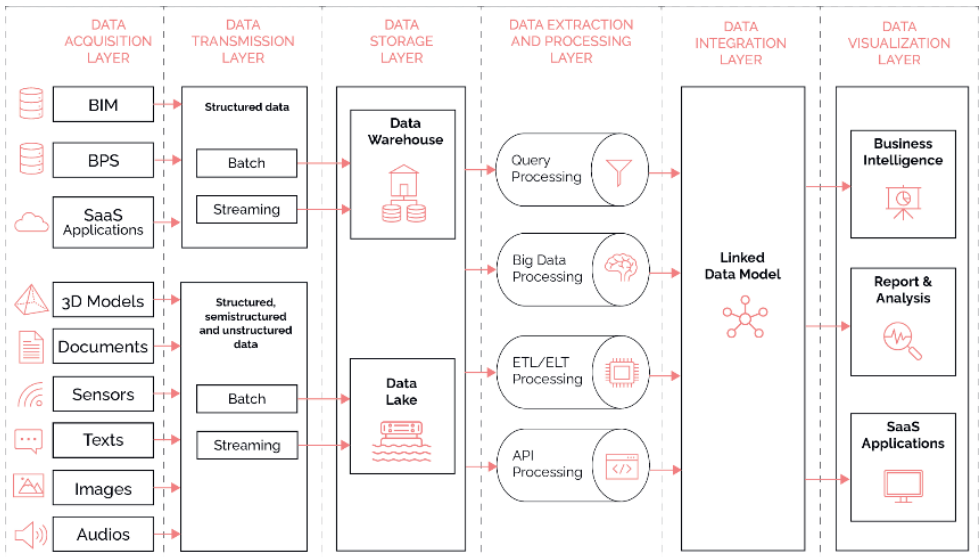
Because of data heterogeneity, many DT implementations

adopt the *hybrid storage approach* described earlier in the text: keep the KG for what it does best (semantic relationships and context) and use specialised stores for what they do best (time-series, big data).

For example, Merino et al. (Merino *et al.* 2022) developed a DT for HVAC fault detection in an educational building by using a KG for the static relationships (building topology from IFC, HVAC system structure from Brick) and a data lake for storing the sensor time-series; the two were linked via unique equipment and sensor identifiers. This allowed real-time data to be ingested without overwhelming the graph database, while still allowing the reasoning engine or query mechanism to understand how the data relates to the building context. Another implementation integrated BIM, BMS, and IoT data by using an RDF graph for static data (IFC and Brick ontologies) and a MongoDB for high-volume sensor data, thereby overcoming performance limitations of RDF stores when handling millions of time-stamped readings (Chamari *et al.* 2023).

In both cases, the static and dynamic parts of the twin remained connected. The KG holds references to the time-series records, ensuring nothing is lost in translation.

Figure 16. Reference system architecture of a DT in the built environment. Source: (Massafra 2024)



2.4.3 Emerging patterns

In summary, across various case studies and applications in the literature and industry, several recurring patterns can be identified. A brief list is provided to open the floor for the following chapters, which will introduce the methodological foundations of 'Buildings as Networks'. These are:

- *KG as semantic backbone, time-series in external store.* static knowledge (building structure, system relationships, metadata) resides in graph databases, while dynamic sensor data resides in a time-series database or cloud data lake.
- *Ontology federation for multi-domain integration.* By federating ontologies and data in graphs, the domain's data can be expanded or updated independently without overhauling the entire knowledge model.
- *Use of standards and open tools.* Many applications rely on open-source graph databases and standard formats to maximise interoperability and avoid vendor lock-in.
- *Connectors and APIs.* Successful deployments often hinge on a robust set of connectors (to get data in/out of the KG) and APIs that expose the integrated data to other services.
- *Incremental adoption via pilot projects.* Many research projects start small – e.g., creating a KG for one building or one use case – then scale up.

The issues outlined here lay the groundwork for more detailed explorations in subsequent chapters of this book, which will delve into specific case studies of graph applications in built heritage and beyond.

Notes

¹ The choice between LPG and KG may depend on the use case. In general, LPGs focus on efficient data storage and traversal, utilising nodes and edges with key-value properties. On the other hand, KGs emphasise semantic meaning, and reasoning through formal models such as RDF triples and ontologies. For these reasons, LPGs are often chosen for real-time analytics and applications with schema-optional models. In contrast, KGs are better suited for data integration, publishing, and complex reasoning across diverse datasets.

² As we will see in some demonstrations in later chapters of the book, in this context, graph algorithms can then compute shortest paths for egress or analyse connectivity to identify critical nodes in an evacuation system, providing critical information both in building design and management.

³ This exponential growth in building ontologies raises a paradox: ontologies are designed to promote the standardisation and homogenisation of knowledge; however, unchecked proliferation could fragment the domain. Since each new ontology effectively establishes its own language, the risk is that it standardises only the workflows of its authors, thereby creating barriers to shared communication across the sector.

⁴ A Uniform Resource Identifier is a unique sequence of characters that identifies an abstract or physical resource. Examples of URIs include a web address (URL), a document, an email address, a book's ISBN code, or a telephone number with an international dialling code.

⁵ Many building ontology files (BOT, Brick, etc.) are published in Turtle (.ttl) format.

⁶ The data model used by Neo4j consists of nodes and relationships, both of which may have properties. Nodes represent entities (with labels and attributes), and relationships represent connections (with types and attributes) between nodes. Labels on nodes help categorise nodes (similar to types in relational tables) and improve query performance.

⁷ For example, by writing a SHACL shape, it is possible to check: 'if every bot:Building has at least one bot:Storey', or 'if an instance is of type brick:Temperature_Sensor, it must have a brick:hasPoint relationship to a brick:Equipment', and so on.

⁸ A KG should ideally record which version of an ontology it conforms to, as definitions might change.

⁹ One of the most comprehensive definitions is the one proposed by the BIM Dictionary (BIM Dictionary 2020), which defines the DT as a set of digital assets consisting of models, documents, and data that represent a physical asset for all or part of its lifecycle. Thus, these are typically three-dimensional models -for example, building models-, rich in data -both static and dynamic- that can enable the physical twin - the existing building- to be known, represented, monitored, and, in some cases, transformed.

¹⁰ As with any novel concept, there are still different definitions and interpretations of DTs. On the surface, a DT might appear to be the same as any BPS or ML model because it is a software representation of some physical entity. Another way to think about a DT in the current literature is as something quite similar to a BIM (de Wilde 2023). As a consequence, while DT is widely regarded as a promising innovation, it has also faced criticism for its sometimes unclear objectives regarding optimisation and efficiency.

METHODOLOGICAL FRAMEWORK

3. FEDERATING ONTOLOGIES FOR CROSS-DISCIPLINARY KNOWLEDGE MODELLING

Abstract

Building on the background exposed in the preceding section, this chapter introduces the research methodology proposed by ‘*Buildings as Networks*.’ In particular, the text presents an *ontology-federation* methodology as the core of a cross-disciplinary Knowledge Graph (KG) framework for building performance management systems.

First, the federation approach is presented as an effective way to address the AECO sector’s data integration gap, along with its rationale, advantages, and challenges. Then, the chapter describes the set of reference ontologies selected for the research, spanning building geometry and topology, building systems, sensor observations, and performance metrics, and explains how these are integrated. Next, a Federated Ontology Framework (FOF) is depicted, highlighting how thematic modules align classes and relationships across sub-ontologies to represent building performance information comprehensively. Finally, the chapter discusses how the FOF is serialised using JSON-LD syntax, coupled with Labelled Property Graph (LPG) technology to build semantically robust storage systems.

3.1 Understanding ontology federation

3.1.1 Definition and rationale of ontology federation

One particular challenge for AECO is the domain's vast, heterogeneous scope. Indeed, the lifecycle of buildings involves several aspects: architectural design, layout features, structures and construction, equipment systems, smart infrastructures, uses, activities and occupancy, energy, comfort, maintenance and renovation, environmental issues, and more. As a direct consequence of the vastness of the domain, no single ontology exists that covers all its facets¹.

A practical solution for modularly representing issues spanning such a wide domain and enabling the creation of cross-disciplinary and multiscale knowledge systems is *ontology federation*. This term refers to the process of connecting and integrating multiple ontologies – each developed independently, with distinct schemas and vocabularies, and covering a specific scope – into a unified, coherent framework (Figure 17).

Instead of attempting to create a monolithic 'all-encompassing ontology' covering every aspect of the built environment, federation provides a strategy for reusing already existing specialised ontologies in concert. The federation produces a *federated ontology*, which, in essence, is a high-level schema that serves as a conceptual model for linking domain-specific sub-ontologies to represent a knowledge domain jointly. This aligns with the general graph principle of flexibility and heterogeneity, where different subdomains contribute to building knowledge with their node and relationship types, and the graph links them as parts of a wider, interoperable system².

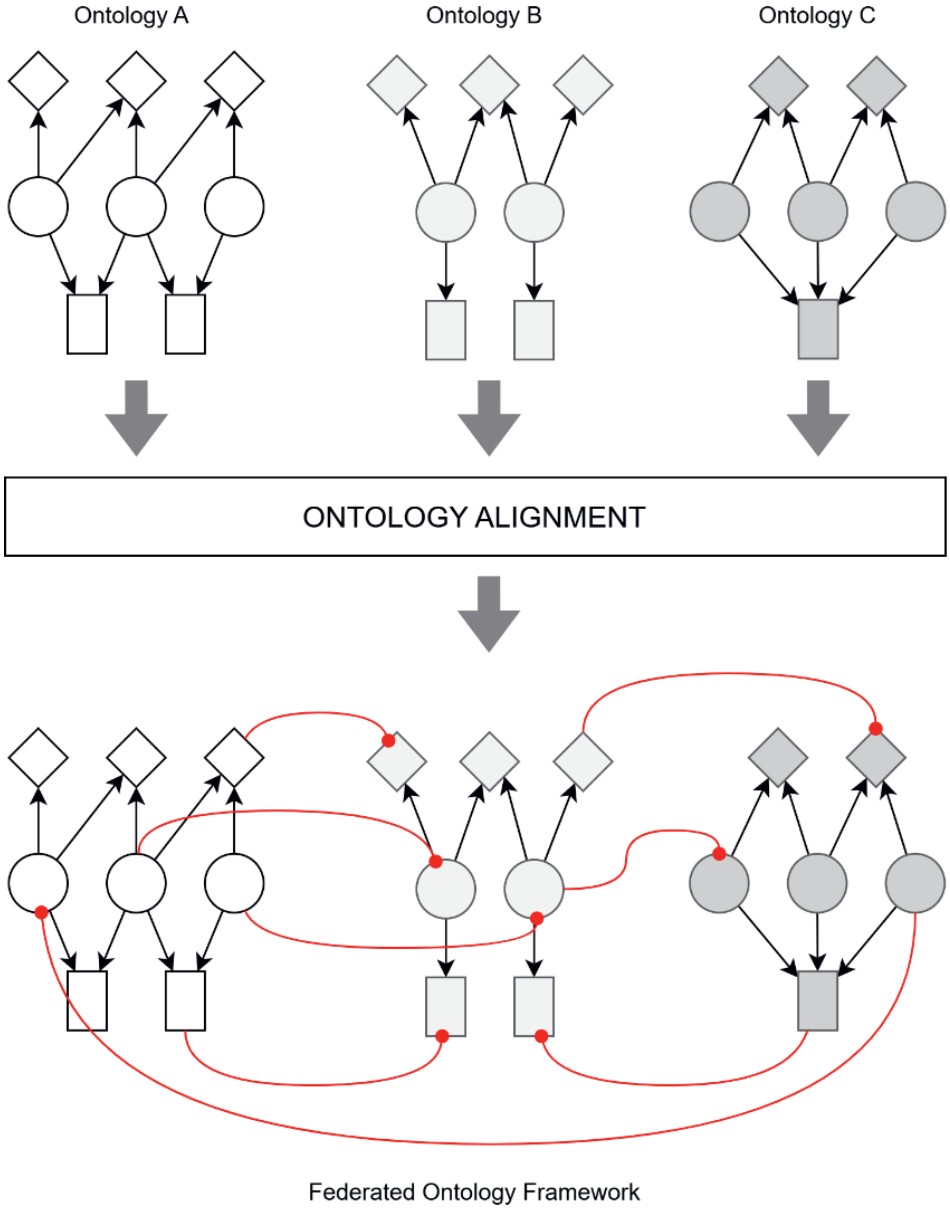


Figure 17. Ontology federation.

3.1.2 Advantages and challenges of ontology federation

Federating ontologies carries several key advantages and challenges for knowledge modelling in complex domains.

Starting from the advantages, they can be summarised in:

- modularity and autonomy;
- semantic interoperability;
- scalability and extensibility;
- reduced redundancy;
- cross-domain consistency.

First, because each domain ontology remains self-contained within the federation, ontology federation promotes *modularity and autonomy*, allowing subontologies to preserve internal consistency and domain experts to continue evolving them independently. The federated model references subontologies rather than rewriting them, so updates to a subontology can be incorporated without redesigning the whole system.

Second, by explicitly mapping relationships between heterogeneous data models, federation promotes *semantic interoperability*, enabling knowledge to flow across subdomains. Aligning concepts through explicit labels ensures that data from different models and databases can be understood in relation to each other, which is crucial for integrated building management.

Third, ontology federation promotes *scalability and extensibility* because new ontologies (and, by extension, new data sources or domains) can be integrated into the KG over time with minimal impact on existing components. The extension of graph databases does not require a complete redesign to accommodate additional domains, making the federation approach future-proof as the scope of building data expands.

Fourth, by leveraging existing ontologies developed by well-established standards bodies and research communities, the federation reduces redundancy, i.e., it prevents redundant modelling efforts. When starting a project, rather than inventing a new schema that covers every aspect (with the attendant risk of inconsistency), the methodology reuses existing vocabularies, which accelerates development and grounds the KG in community-vetted standards.

Finally, by declaring which classes in different ontologies correspond to each other and how data properties relate, federation ensures *cross-domain consistency* through explicit semantic

alignment, resulting in a unified view of information in which queries can traverse relationships between different types of building data.

Despite its benefits, ontology federation introduces some challenges that must be carefully considered before embarking on such a path.

Designing and maintaining the alignments between ontologies can be complex and labour-intensive, and it requires a good understanding of the ontologies to be federated. Each ontology has its own structure and logic, so identifying equivalent or related concepts and deciding how to connect them demands both domain knowledge and ontology engineering skills. Given the great variety of ontologies and the different purposes for which they have been built, ensuring that these mappings are correct and unambiguous is non-trivial. There is often an alignment overhead and, sometimes, bridging classes, properties, or rules need to be created to reconcile differences or overlaps between subontologies³.

In addition, a federated model can become complex to maintain. Changes in any of the subontologies (e.g., new versions or extensions) may require updates to the federation alignments. This necessitates ongoing governance: monitoring ontology development, managing version control, and potentially adopting an upper-ontology or a common alignment strategy to handle how terms from different sources relate. Without careful governance, inconsistencies can arise as ontologies evolve.

Finally, exceptionally complex federations lead to poorly accessible knowledge systems. The more complex the federated ontology network, the steeper the learning curve is for those who need to use or extend it. A right composition should be found when building a federated ontology that should, on the one hand, be as simple and understandable, and on the other, sufficiently accurate not to constrain domain modelling.

3.1.3 Methodological process for federating ontologies

Creating a federated ontology involves a systematic process, typically including the following steps:

1. Ontology selection;
2. Alignment design;
3. Mapping and bridging axioms;
4. Integration and testing.

In the first phase, *ontology selection*, suitable reference ontologies for each knowledge domain relevant to the problem representation are identified. Criteria for selection include domain scope, maturity, community adoption, compatibility with other ontologies, accessibility, and good documentation. For a building performance management graph, for instance, the chosen ontologies should cover areas such as building topology (spaces and elements), building systems (equipment, HVAC), sensors and observations, and (but not only) performance metrics. Using an ontology lookup service or library (e.g. the Built Environment Ontology Lookup Service) can aid discovery in this step.

In the second phase, *alignment design*, the selected ontologies are analysed to identify points of intersection or overlap and determine how they can be connected. Key alignment tasks include mapping equivalent classes (e.g. recognising that an 'IfcSpace' in ifcOWL corresponds to a 'bot:Space' in BOT) and defining how instances from different ontologies relate (e.g. a Brick sensor instance should link to a BOT space instance representing its location). This may involve drawing conceptual alignment diagrams or matrices. In this step, it is crucial to review and study the documentation for each ontology carefully.

In the third phase, *mapping and bridging axioms*, the alignments are technically implemented by extending or bridging the ontologies. This typically means writing OWL axioms or SHACL rules to link concepts. Bridging can be achieved via `owl:equivalentClass`, `owl:equivalentProperty`, `rdfs:subClassOf` or custom object properties that connect individuals across ontologies. For example, one might introduce a property `hasLocation` that links a piece of equipment (from Brick) to a spatial location (from BOT) if such a relation is not already present in the federated ontologies. These custom classes and properties can be stored in a separate 'bridge ontology' explicitly created to fill gaps⁴.

Finally, in the last phase, *integration and testing*, once mappings are in place, sample data is integrated from each domain to populate a KG and test the federated ontology. This involves serialising data in a graph-compatible format and linking data from different sources (e.g., BIM files, sensor databases, simulation outputs) to the corresponding ontology classes. For testing, running queries that traverse multiple domains helps verify if the federation works. Creating prototypes and testing them for specific information needs can also help identify bugs or gaps in the federated ontology. This step often reveals misalignments or missing links, which can then be corrected in the ontology design.

3.2 Reference ontologies for building performance management

3.2.1 Rationale behind ontology selection

As mentioned, this book focuses on performance-based building management. The research approach adopted here implemented a Federated Ontology Framework (FOF) to manage the performance aspects of buildings during the operational phase.

To implement the FOF, a set of reference ontologies was chosen for their relevance and complementarity. The ontology selection process led to a set of subontologies, each originating from a different community or standard and serving a distinct role within the federated model.

This section serves a dual purpose: on the one hand, it presents the outcome of the ontology selection phase used to construct the FOF; on the other hand, it functions as a concise literature review, introducing various ontologies relevant to performance management.

3.2.2 Industry Foundation Classes and ifcOWL

The *Industry Foundation Classes (IFC)* is an open, international standard for BIM (ISO 167391:2024). The schema was developed and is maintained by buildingSMART, a worldwide industry body driving the digital transformation of the industry.

IFC draws on decades of standardisation efforts to transform the construction industry through IT. It offers a vendor-neutral, platform-agnostic data model that codifies the identity, attributes and relationships of building elements, supporting the digital representation of physical components (e.g., walls, columns, mechanical systems), spatial elements (e.g., buildings, storeys, and spaces), but also abstract concepts (e.g., costing) and processes (e.g., tasks or operations).

IFC models rely on IFC terminology. In IFC, an entity refers to any concept, whether abstract or tangible, characterised by constraints and attributes. An attribute, instead, is the smallest unit of information for each entity. Among the various classes offered by IFC, the `element` class is fundamental, as it allows the representation of building components, their geometry, their associations with other objects (`IfcRelationship`), and their attributes and properties (`IfcPropertyDefinition`). In an IFC model, each instance corresponds to a specific entity that inherits semantics and data from the supertype `IfcElement`. Such instances, considered synonymous with *object* and *occurrence* (`IfcObjectDefinition`), are associated with one or more *properties* and serve as units of information⁵.

The schema is currently serialised in the EXPRESS⁶ language and organised in a taxonomic hierarchy of generalisation and specialisation. New serialisations are emerging to make IFC more accessible and interoperable with web and modern software ecosystems, like IFC JSON (buildingSMART International 2025). The schema can also be represented in Web Ontology Language (OWL) using `ifcOWL`, which enables the use of semantic web and Linked Data (LD) technologies.

Although the schema is internationally recognised and encompasses a wide range of content, some challenges limit the extent to which IFC's rich semantic data can be exploited in downstream applications (Xie *et al.* 2021).

As a consequence, IFC is still not fully exploited in practice. Its broad scope often leads to excessive complexity, creating accessibility barriers for BIM practitioners in the field. For example, not all IFC entities and classes are equally used in practice, partly because BIM software does not fully support the entire IFC schema. The export process from native software remains challenging, while ambiguities in data modelling and limited

interoperability with other tools persist (Antunes *et al.* 2024).

Moreover, the schema alone does not address how to handle dynamic data, such as IoT streams. Therefore, to build modern applications such as digital twins (DTs), it is necessary to combine it with more specialised schemas.

3.2.3 Building Topology Ontology

Given the complexity of the IFC schema, the need for more streamlined and flexible ontologies emerged. For this reason, the World Wide Web Consortium (W3C) established the Linked Building Data (LBD) community group, a set of experts in BIM and Web of Data technologies. One of the group's widely known outputs is the *Building Topology Ontology (BOT)*, introduced by Rasmussen *et al.* (Rasmussen *et al.* 2017).

Before releasing BOT, Rasmussen and colleagues observed that, in recent years, many ontologies were developed within the AECO, often redefining fundamental concepts already present in the publicly available IFC schema. According to their opinion, this issue was due to the high complexity of the IFC schema, conflicting with the W3C principle that ontologies should remain simple to ensure ease of use and maintenance.

Given the recurrence of basic spatial and topological concepts across ontologies not explicitly intended for such purposes⁷, the BOT's group proposed a simplified ontology focused exclusively on core building spatial concepts. This ontology is centred on building *topology*, capturing both physical and conceptual components and their interrelationships. BOT's design pursues two objectives: first, it enables the use of a concise set of fundamental building classes; second, it allows targeted extensions towards specialised domain ontologies governed by industry experts or standardisation bodies.

In its initial version, BOT provided a foundational framework based on the hierarchical decomposition of a building into storeys and spaces (Figure 18). Over time, this framework has been further refined and is now formally documented by the World Wide Web Consortium (W3C).

At present, BOT is structured around seven principles:

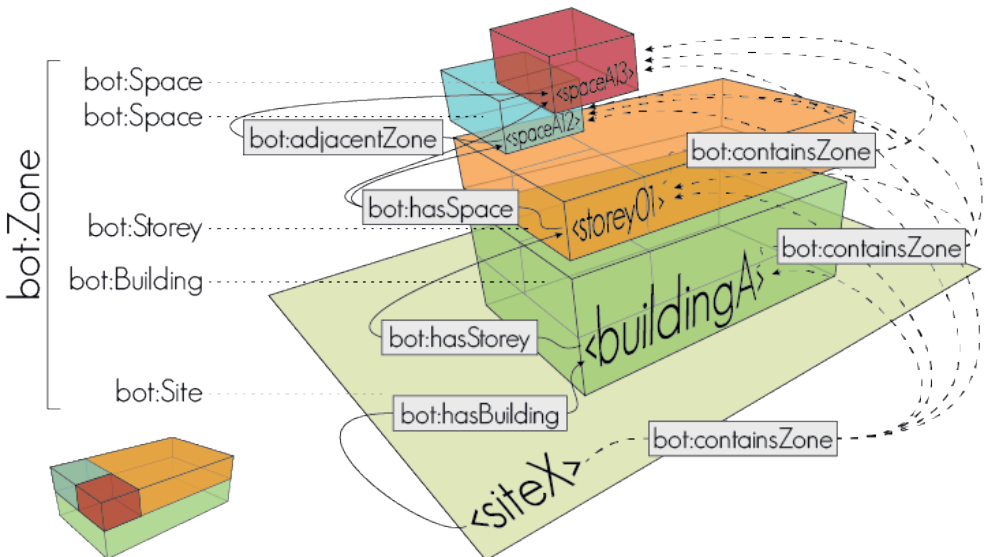
1. Zones are defined as three-dimensional spatial volumes, encompassing Buildings, Storeys, and Spaces;
2. Zones can contain other zones, with Buildings containing

- Storeys, and Storeys containing Spaces;
- 3. Zones may intersect or be adjacent to other zones;
- 4. Building elements may consist of sub-elements;
- 5. Zones may include Elements, whether contained, adjacent, or intersecting;
- 6. Adjacent zones and/or elements share an Interface;
- 7. Both Zones and Elements are associated with a 3D model (geometry, materials, etc.).

BOT formalises this structure through seven classes (bot:Building, bot:Element, bot:Interface, bot:Site, bot:Space, bot:Storey, and bot:Zone) together with fourteen object relationship types⁸.

Due to its simplicity, BOT has become a widely adopted foundational ontology, offering a robust basis for integration with more specialised ontological frameworks⁹.

Figure 18. Classes and relationships involved in BOT's zones. Source: Building Topology Ontology Draft Community Group Report of the 28th of June 2021 (W3C 2021)



3.2.4 *Topologic's Class Hierarchy*

Developed by a research team from Cardiff University and University College London under the guidance of Prof. Wassim Jabi, *Topologic* is a computational framework for building spatial analysis that establishes a formal class hierarchy to represent spatial and topological aspects of architecture (Figure 19) (Jabi and Chatzivasileiadi 2021).

Although not explicitly conceived as an ontology, this hierarchy introduces abstract concepts such as `top:CellComplex`, `top:Cell`, `top:Face`, `top:Edge`, `top:Vertex`, and `top:Aperture`, which characterise a building in terms of volumes (spaces), surfaces (walls, slabs, roofs), openings (doors, windows), and their associated topological relationships, including adjacency and connectivity.

The hierarchy is operationalised through *Topologic* (Jabi and Chatzivasileiadi 2021) and *Topologicpy* (Jabi 2024). This software library enables the representation of buildings as topological graphs, with a particular emphasis on architectural shape grammar, space syntax, and, more broadly, spatial reasoning.

Topologic's hierarchical structure, combined with its corresponding software, enables the operationalisation of graph representations of buildings. Defining multiple classes that represent building topology through abstract yet interconnected geometries facilitates rule-based query operations for modelling and retrieving building information using ad hoc methods. For instance, in *Topologic*, users can query the faces, edges, or vertices of a cell, with the system returning distinct, non-overlapping lists for each category. These entities can then be further interrogated: a `Cell` can provide its `Faces`, each `Face` can provide its `Edges`, and each `Edge` can return its `Vertices`. Notably, the framework also supports ascending queries. For example, a `Cell` may be queried for its `Vertices`, which can then be used to identify the `Edges` they are associated with, or a set of `Edges` may be examined to determine the `Faces` they connect to. This bidirectional querying supports conceptual and practical modelling by integrating topological operations with other semantic processes.

In 2024, preliminary efforts were undertaken to align *Topologic* with *BOT*, as the two frameworks are inherently compatible. The distinctive contribution of *Topologic* lies in its combi-

nation of a formalised class hierarchy with a functional toolkit, enabling the operationalisation of topological relationships within digital building models. Figure 20 presents a synthesis of this integration effort.

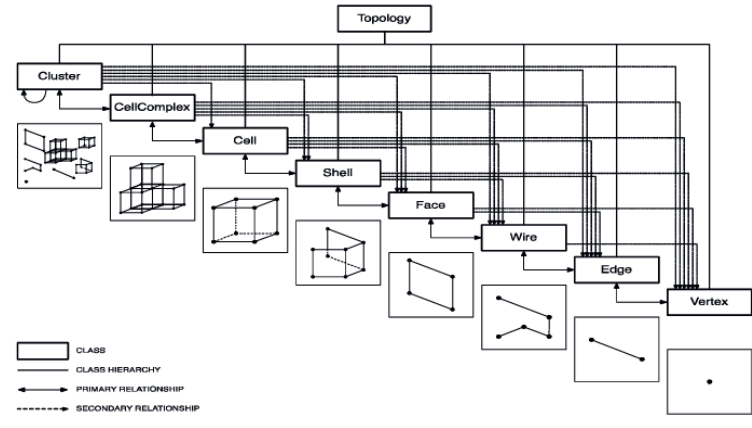


Figure 19. Topologic's class hierarchy. Source: (Jabi and Chatzivasileiadi, 2021)

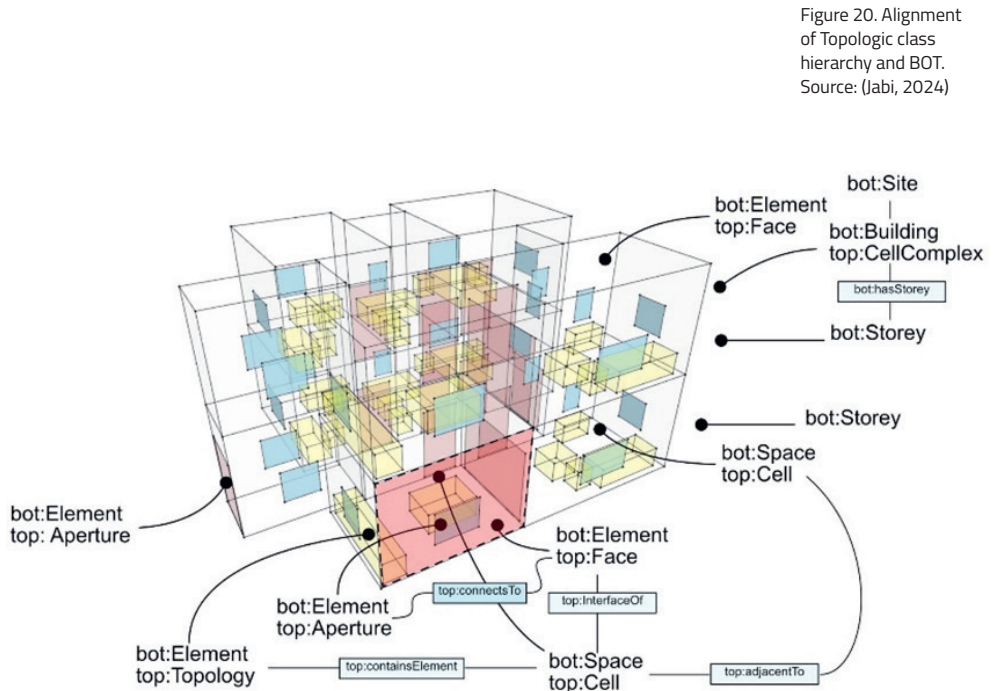


Figure 20. Alignment of Topologic class hierarchy and BOT. Source: (Jabi, 2024)

3.2.5 Brick Schema

In specific contexts, the IFC format proves inadequate for providing a comprehensive representation of all building components, particularly MEP/HVAC systems, when explicit geometric references are absent (Xie *et al.* 2021). Components such as HVAC devices, sensors, and meters can be more effectively captured through alternative ontologies, among which Brick is a notable example.

Developed by Balaji and colleagues (Balaji *et al.* 2018), Brick is an open-source initiative that aims to deliver a unified ontology for describing equipment, sensors, control systems, and their interconnections, which are particularly relevant to building operations mapping. Leveraging semantic web technologies, Brick can systematically represent the heterogeneous characteristics, resources, and subsystems found across a broad spectrum of building types.

Through its specialised vocabulary, Brick offers greater flexibility than IFC in representing complex hierarchies and interdependencies across building systems. This includes aspects such as taxonomy, location, equipment connections and composition, point associations, and monitoring functions.

The ontology's class hierarchy is structured around the central concept of *Point*, which represents physical or virtual entities within Building Management Systems (BMSs) generating time-series data. The *Equipment* and *Location* classes complement this core class. The relationships established among points, their associated equipment, and the spatial contexts in which they are embedded define the operational behaviour of building systems and equipment, as well as their influence across different zones of the building. Figure 21 and Figure 22 illustrate an overview of these relationships.

In its initial applications, the Brick framework was tested on more than 17,700 data points derived from BMSs across six buildings, demonstrating its applicability to systems such as lighting, power, water, heating, ventilation, and air conditioning (Balaji *et al.* 2016, 2018). Since then, numerous implementations have been developed. For example, Xie *et al.* combined Brick with IFC to track sub-metering systems and sensors in a UK educational facility, as part of one of the first DT frameworks in the field (Xie *et al.* 2021).

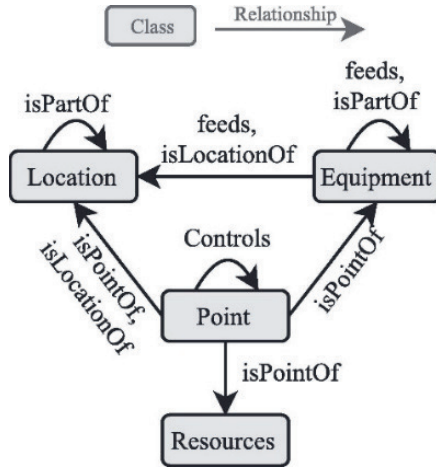


Figure 21. Central concepts in Brick and their relationships. Source: (Balaji et al. 2018)

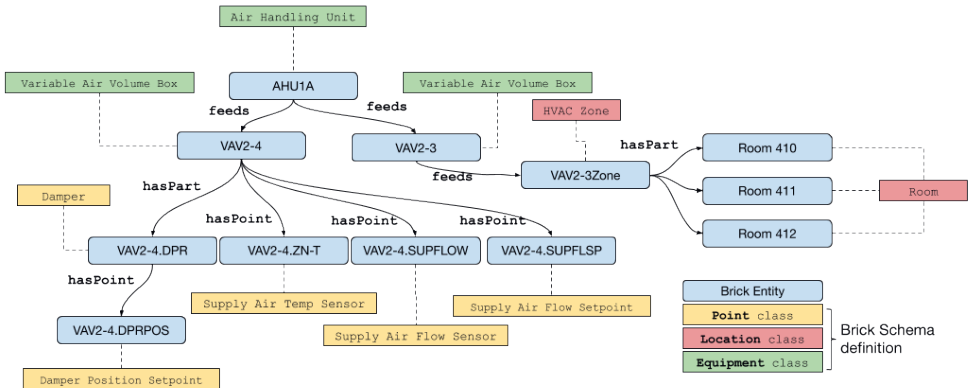


Figure 22. Example of Brick graph representation. Source: (Balaji et al. 2018)

3.2.6 SSN and SOSA ontologies

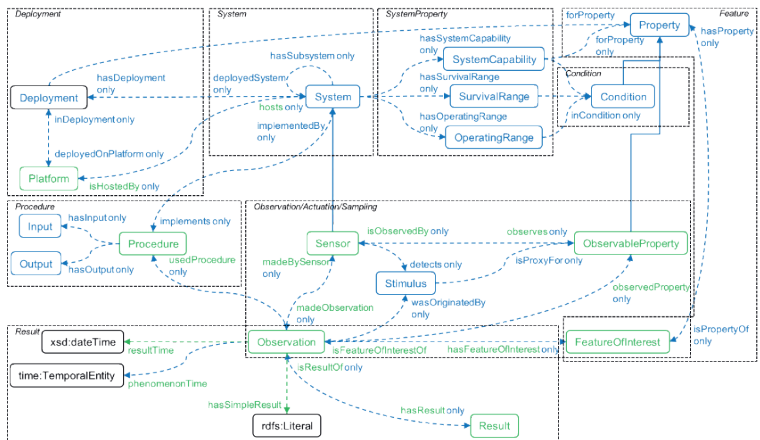
In addition to ontologies that describe the topological aspects of spaces, elements, and systems, there are specialised ontologies for modelling time-series data, such as sensor data. Notable examples include the *W3C Semantic Sensor Network (SSN)* ontology and its lightweight counterpart *Sensor, Observation, Sample, and Actuator (SOSA)*. Together, they provide a framework for describing sensors, the observations they produce, and related concepts.

Initially developed by the W3C SSN Incubator Group, SSN

offers a detailed model of sensors, their operational procedures, outputs, and the features of interest they monitor (Compton *et al.* 2012). SOSA, by contrast, constitutes a simplified and reusable core ontology, defining essential classes such as `sosa:Sensor`, `sosa:Observation`, `sosa:ObservableProperty`, `sosa:Actuator`, as well as relations including `sosa:observes`, `sosa:madeBySensor`, and `sosa:hasResult` (Janowicz *et al.* 2018). Figure 24 presents an extract of SSN classes and properties, focusing on the observation perspective.

Although SSN/SOSA’s subdomain may appear to overlap with Brick, the two ontologies are complementary. While Brick focuses on identifying the sensor network’s topological structure and mapping spaces, devices, and what they measure, SSN/SOSA is used to represent sensing and the resulting measurements¹⁰. Thus, a specific measurement, like a temperature reading at a specific datetime, can be modeled as an instance of `sosa:Observation`, linked to a `sosa:Sensor` – which would also correspond to a `brick:Sensor` – but also to a `sosa:ObservableProperty` such as ‘air temperature’. In this way, SSN and SOSA provide a semantic framework for handling time-indexed observational data and connecting it to the building context.

Figure 23. Overview of the SSN classes and properties (observation perspective). Source: (Compton *et al.* 2012)



3.2.7 Ontology for managing KPIs

Sensor and simulation data are valuable for performance management; however, since they often consist of large volumes of raw data, they need to be synthesised into relevant KPIs that are more meaningful and explanatory¹¹.

Ontologies for such a purpose exist as well.

Developed by the Ontology Engineering Group at Universidad Politécnica de Madrid within the scope of the BIMERR project, the *Key Performance Indicator Ontology (KPIO)* was created to formalise performance indicators in buildings (Feria and Villalón n.d.). It provides a structured vocabulary for defining KPIs, including their formulas, measurement units, thresholds, and relationships with building elements or systems. Among its key classes are `kpi:KPI`, `kpi:KPIValue`, and `kpi:Scenario`, which enable indicators to be linked to specific data sources and contextual information (Figure 24).

Although initially designed for energy efficiency assessment, KPIO can be conceived as a generic framework for performance evaluation and is sufficiently flexible to be extended to other domains, such as thermal comfort, maintenance, or cost analysis. Furthermore, it supports the representation of `Scenario` objects, which allow KPIs to be semantically associated with both actual and hypothetical condition states. This provides a consistent way to describe which performance metrics are being evaluated and under what circumstances, orienting DSS towards scenario-based, multicriteria analysis.

In addition to KPIO, the *Energy Management Key Performance Indicator Ontology (EM-KPI)* was developed to support the exchange of multi-level energy performance information. It introduces a set of concepts for energy management at both the district and building scales, encompassing key domains such as building characteristics, weather conditions, occupancy, utilities, and energy-related KPIs (Li *et al.* 2019).

EKO defines classes similar to KPIO, such as `eko:KPI` and `eko:KPIValue`, which align with the concepts introduced in the KPI ontology, while also extending the model through additional relationship types – for instance, `eko:hasAssociatedObject` – that can enable KPIs to be linked to specific objects and organised into groups (Figure 25).

This relational structure allows EKO to represent energy

3.3 Implementation of a federated ontology framework

3.3.1 *The need for a FOF*

Table 5 provides a comparative overview of the coverage offered by each ontology discussed in the previous paragraph across various aspects of building data, namely geometry, topology, systems, sensors, and KPIs.

As previously noted, since each ontology addresses only a subset of these representational needs, a Federated Data Modelling (FDM) approach was adopted in this research to develop a Federated Ontology Framework (FOF) that encompasses all these aspects through the integration of federated sub-ontologies. This approach avoids redundant redevelopment of existing ontological structures, leveraging the extensive efforts of practitioners and researchers who have developed these sub-ontologies over the past decade.

In doing so, the FOF aims to provide a comprehensive representation of the building performance domain, serving as the semantic schema for creating cross-disciplinary KGs, enabling unified queries and analyses over data ranging from BIM models to BPS models to sensor readings and related performance metrics. In practice, the FOF establishes the semantic core that underpins the applications presented in the following section of this book, spanning several subdomains of building management, including spatial analysis, fire safety assessment, micro-climatic monitoring, and energy and occupancy management.

3.3.2 *Structure and core modules*

In line with FDM principles, the FOF was designed with a modular structure. In this structure, the mentioned ontologies contribute, in varying proportions, to the formation of thematic modules, each corresponding to a specific aspect (subdomain) of building performance information.

Figure 26 and Figure 27 illustrate the overall architecture of the FOF, which is conceptually divided into distinct modules. The first figure presents the classes and relationships, colour-coded according to the sub-ontology from which they originate (i.e., IFC, BOT, Topologic, SSN/SOSA, Brick, KPIO, and

EM-KPI). The second figure, in contrast, depicts the classes and relationships after federation, colour-coded by module. More specifically, the modules and their purposes are as follows:

- *Spatial Element Module*: represents the spatial hierarchy of elements within the building, serving as the backbone for locating other components.
- *Interface Module*: represents construction elements that delimit spaces, such as envelope and partition elements (e.g., walls, roofs, slabs, openings).
- *Equipment & Sensor Module*: represents the topology of equipment and sensor networks and links their instances to sensor recordings.
- *KPI Module*: represents KPIs, their calculation methods, and values, connecting them to relevant spatial elements.
- *Document Module*: a module that explicitly models the sources used for data modelling (e.g., documents, databases, digital models).
- *Property Module*: attributes properties and data to the spatial, equipment, and interface entities.

Table 5. Comparative coverage of the selected ontologies across different building aspects.

Ontology	Geometry	Topology	Systems	Sensors	KPIs
IFC	Yes	Yes	Yes	Limited	No
BOT	No	Yes	No	No	No
Topologic	Limited	Yes	No	No	No
Brick	No	No	Yes	Yes	No
SSN/SOSA	No	No	No	Yes	No
KPIO	No	No	No	No	Yes
EM-KPI	No	No	No	No	Yes
FOF	Yes	Yes	Yes	Yes	Yes

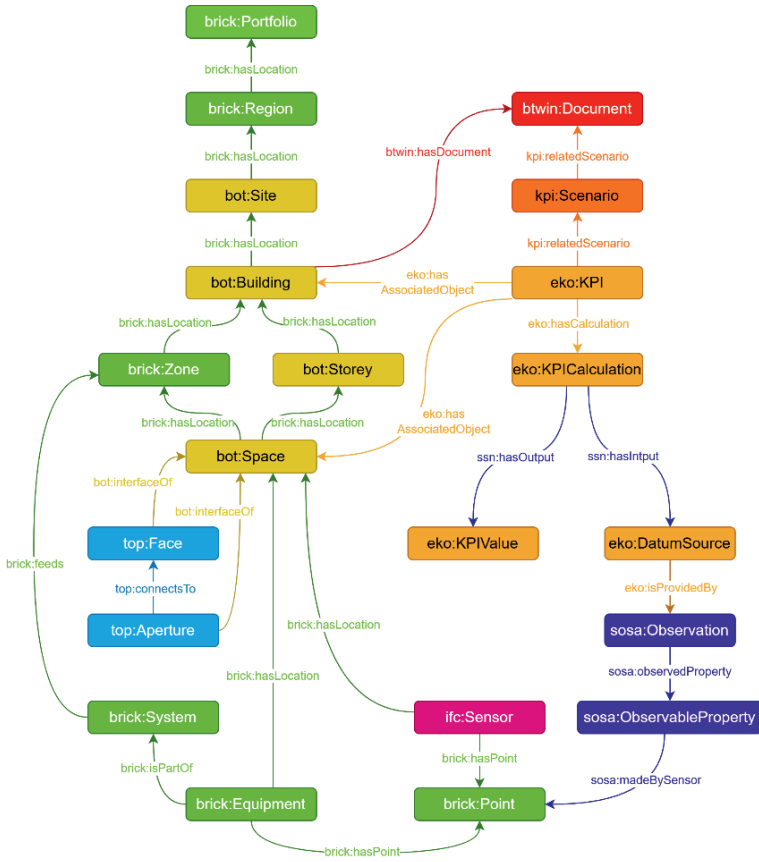


Figure 26. Federated ontology framework for building operational performance. Classes and relationships coloured by subontology.

Legend

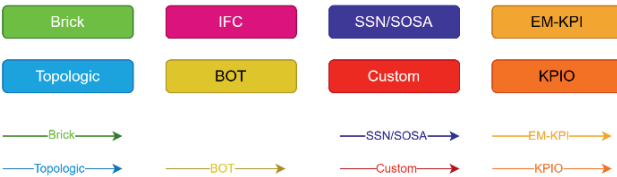
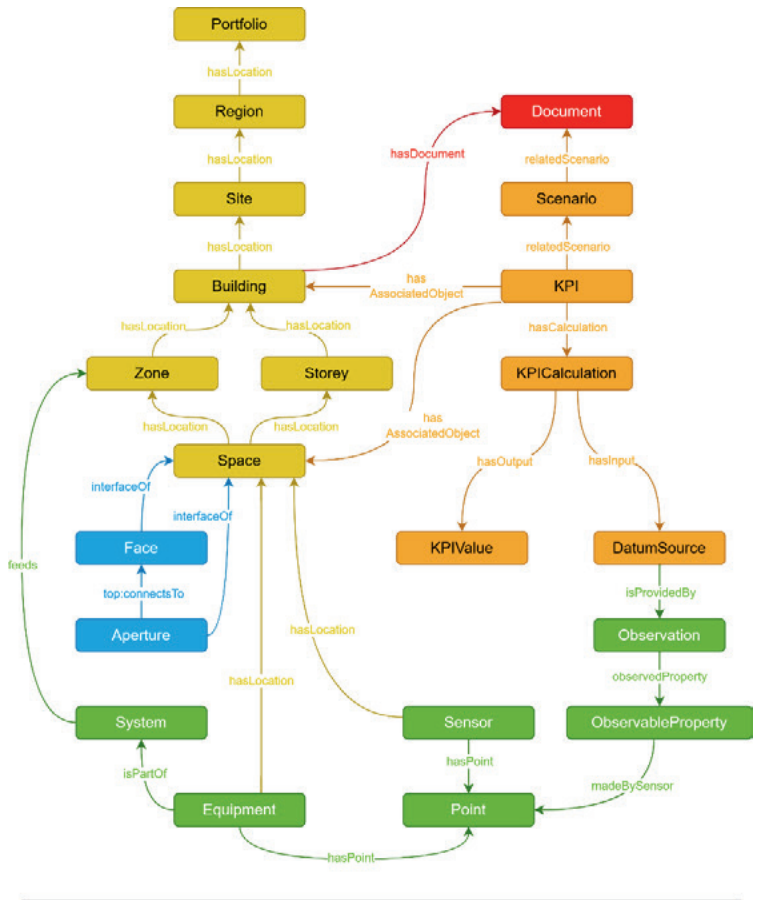
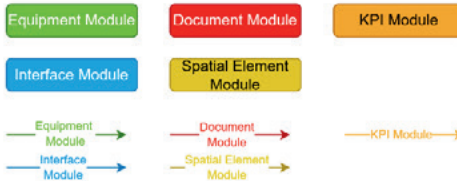


Figure 27. Federated ontology framework for building operational performance. Classes and relationships coloured by module.



Legend



3.3.3 Spatial Element Module

The *Spatial Element Module* captures the core spatial elements and their hierarchy.

It combines classes from BOT, used for referencing fundamental spatial containers, with a few extensions from Brick for higher-level grouping.

Key classes from BOM include `bot:Site`, `bot:Building`, `bot:Storey`, `bot:Space` for the central building hierarchy. In addition, to allow grouping of spaces by functional criteria, Brick's notions of `brick:Zone`, `brick:Region`, and `brick:Portfolio` are incorporated¹² (Table 6).

All spatial entities can be related through containment connections via the `brick:hasLocation` relationship, which links a space to its storey, a storey to its building, a building to its site, and so on. Figure 28 and Figure 29 provide example graph representation based on the Spatial Element Module.

This layered spatial structure provides also the anchor for all other data. For instance, equipment and sensors can be related to spatial elements through the `brick:hasLocation` relationship.

FOF	IFC	BOT	Topologic	Brick
Portfolio	-	-	-	Portfolio
Region	-	-	-	Region*
Site	IfcSite	Site	-	Site*
Building	IfcBuilding	Building	CellComplex	Building*
Zone	IfcZone	Zone	-	Zone*
Storey	IfcBuildingStorey	Storey	-	Floor*
Space	IfcSpace	Space	Cell	Space*

Table 6. Alignment of Spatial Element classes among the federated subontologies.

* Classes deprecated in the last version of Brick (Brick 1.4)

Figure 28. Example graph illustrating the composition of a university's building portfolio.

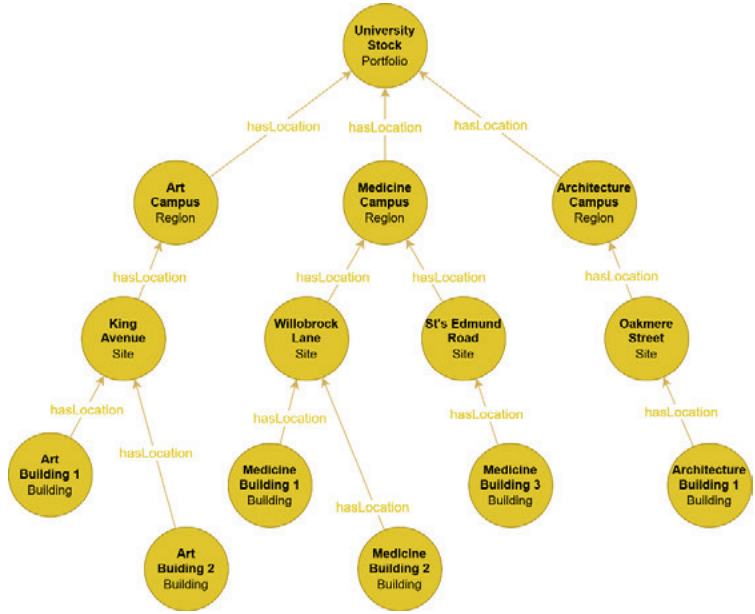
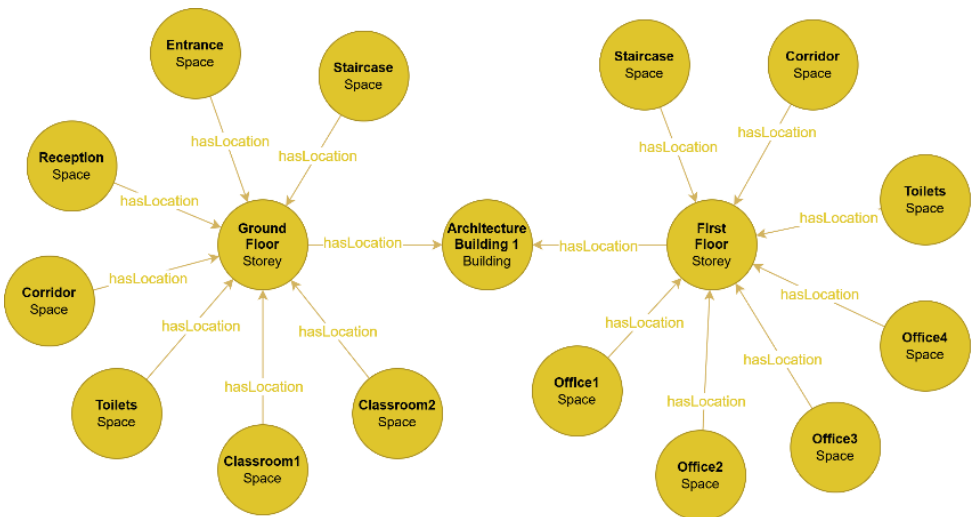


Figure 29. Example graph representing the spatial composition of a university building.



3.3.4 Interface Module

The *Interface Module* enables the representation of building elements that form interfaces between spaces, such as walls, floors, roofs, doors, and windows, whose characteristics are fundamental to the building performance domain for analyses including space topological connectivity, thermal transfer, egress routes, and safety assessment.

Within this module, classes from the Topologic hierarchy are used alongside BOT relationships. Surface elements, like walls, slabs, and roofs, are represented as `top:Face` instances, while opening elements, like doors and windows, are represented as `top:Aperture` instances (Table 7).

BOT's property `bot:interfaceOf` is used to link these interface elements to the two spaces they separate (e.g. a wall as an interface between a space and another). Additionally, the Topologic relation `top:connectsTo` is used to explicitly capture connections between apertures and interfaces (e.g. a door aperture hosted in a wall)¹³.

Below are some illustrative figures of the Interface Module.

Figure 30 shows an example of a graph in which the relationships between spaces and doors are mapped through the `bot:interfaceOf` relationship. Figure 31 shows the same graph, where these relationships are compressed into space-to-space relationships via the custom, interclass relation `btwin:hasPassageTo`. Similarly, Figure 32 presents an example of a graph in which spatial adjacency relationships are mapped according to the space-to-wall logic. Figure 33 shows the compressed version, where the space-to-space adjacency relationships are mapped through the interclass relation `btwin:isAdjacentTo`.

FOF	IFC	BOT	Topologic
Face	IfcWall, IfcSlab, IfcRoof, IfcCurtainWall	Interface	Face
Aperture	IfcDoor, IfcWindow	Interface	Aperture

Table 7. Alignment of Spatial Element classes among the federated subontologies.

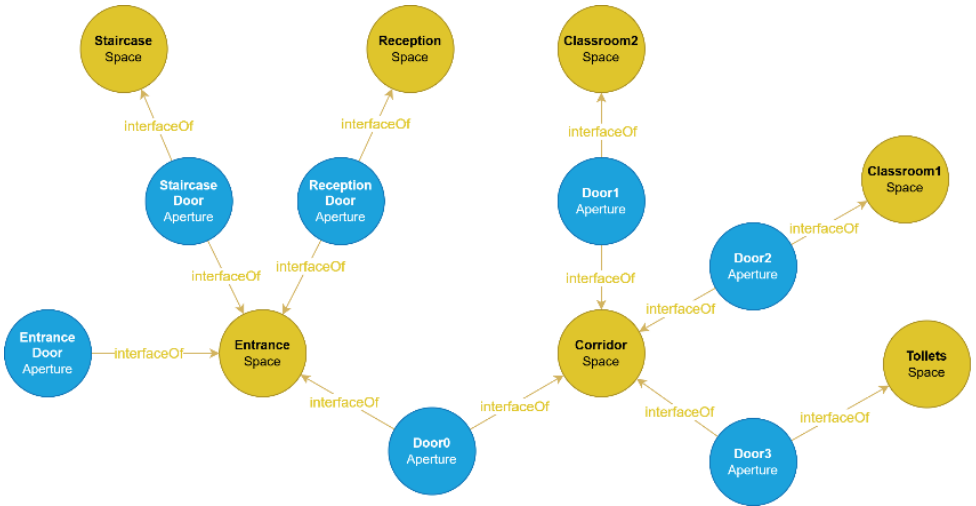
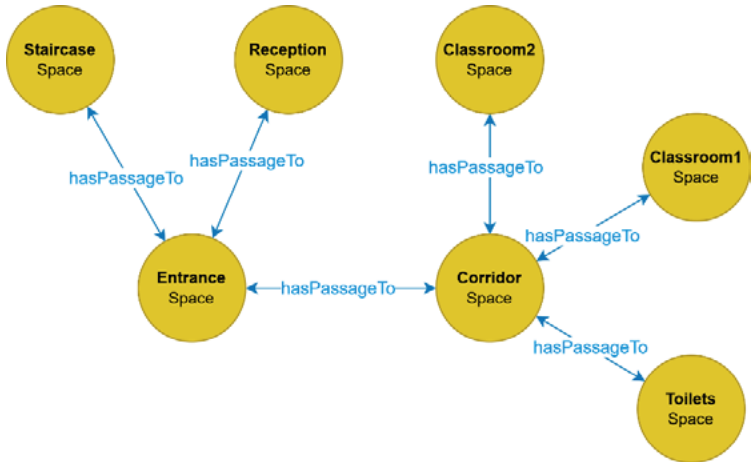


Figure 30. Example graph representing the space-to-door connections of the layout of a storey of an university building.

Figure 31. Example graph representing the space-to-door connections of the layout of a storey of an university building. The space-to-door connections are compressed in space-to-space passage relationships to the `btwin:hasPassageTo` class.



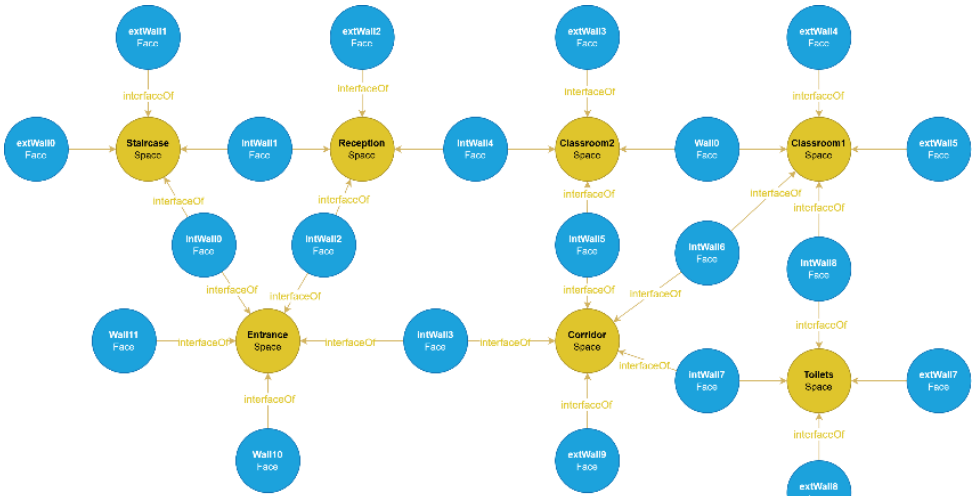


Figure 32. Example graph representing the space-to-wall connections of the layout of a storey of an university building.

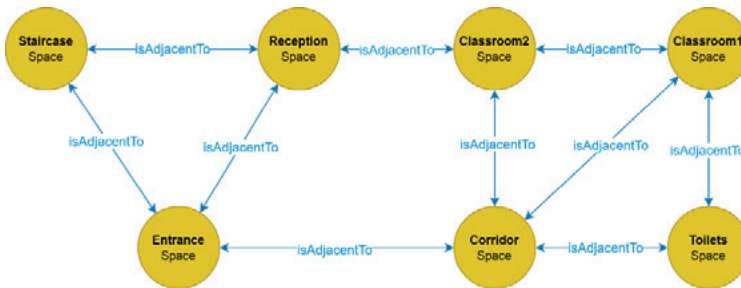


Figure 33. Example graph representing the space-to-wall connections of the layout of a storey of an university building. The space-to-wall connections are compressed in space-to-space passage relationships to the `btwin.isAdjacentTo` class.

3.3.5 Equipment & Sensor Module

The *Equipment & Sensor Module* addresses the representation of building systems, sensing networks, and their associated sensing points and measurements.

It mainly builds upon Brick for the definition of equipment and sensor classes and draws on SSN/SOSA for modelling observations.

Key classes in the module include `brick:Equipment` for equipment devices (e.g., an air handling unit, a boiler, or a fan) and `brick:System`, which allows grouping equipment devices into higher-level sets (e.g. an HVAC system).

Each equipment instance can be associated with one or more `brick:Point` entities, representing sensors, setpoints, or actuators connected to that equipment. For example, a room thermostat would be modelled as a `brick:Point` linked to a `brick:Equipment` (such as an AHU) through the `brick:hasPoint` relation.

The `ifc:Sensor` class is also employed in this module to describe the physical sensor device. In practice, a sensor may be represented simultaneously as an `ifc:Sensor` and a `brick:Sensor` (a subclass of `brick:Point`). The IFC definition intends the sensor as a physical, electronic device located in space with specific technological properties. In contrast, the Brick definition focuses on the type of measurement it performs (e.g., temperature, humidity, or light). As a single physical device may include multiple measurement capabilities, the `ifc:Sensor` corresponds to the device itself, which can be associated with several `brick: -Sensors` (e.g., temperature and humidity).

The `ifc:Sensor` can be also connected to `bot:Space` elements via the `brick:hasLocation` relation, while the `brick:Sensor` is connected back to the `ifc:Sensor` through the `brick:isPointOf` relation. Similarly, `brick:Equipment` instances can be connected to spaces using `brick:hasLocation`, reflecting their role as physical devices within the spatial system.

To capture equipment interconnections – such as an electrical meter supplying a distribution panel, or an AHU supplying a zone – the `brick:feeds` relation can be applied between `brick:Equipment` objects, or between `brick:Equipment` or `brick:System` objects and the spatial objects.

The `brick:Sensor` class is equivalent to the `sosa:Sensor` class

in SOSA. This object is intended to generate `sosa:Observation` instances, which represent the single time-series measurements. The module also incorporates `sosa:ObservableProperty` to define the measurement type (e.g., temperature, humidity, or CO₂)¹⁴.

Figure 34 provides an example of a graph representing the equipment network of a sample building and its relationship with the building's spatial structure. Figure 35, instead, depicts a sensing system within the same sample building.

3.3.6 KPI Module

The *KPI Module* supports performance assessment and scenario analysis. It is primarily based on the EM-KPI and KPIO ontologies. In this module, the `eko:KPI` class represents a KPI definition, while the `eko:KPIValue` class is used for actual measured or calculated KPI values.

In addition to these, the `kpi:Scenario` class allows for representing scenarios, intended as hypothetical projections referring to possible future states (usually used in multicriteria decision-making to explore how different variables and decisions might interact and affect outcomes)¹⁵. `eko:KPI` objects can be related to `kpi:Scenario` objects through the `kpi:relatedScenario` relationship.

The FOF allows linking a KPI to the spatial element it is associated with via `eko:hasAssociatedObject`. For instance, an energy use KPI might be associated with a `bot:Building` (if it measures whole-building energy) or a `brick:Zone` (if it is zone-specific).

Temporal aspects are handled by reusing classes from the *Time Ontology* such as `time:Instant` or `time:Interval` to denote when a KPI value is measured or over what period it applies.

A custom class, `btwin:KPISet`, was also introduced in the FOF to group KPI instances for more convenient handling¹⁶. `btwin:KPISet` is related to `eko:KPI` objects through the custom relationship `btwin:hasKPIs`, to `kpi:Scenario` through `kpi:relatedScenario`, and to spatial elements through `eko:hasAssociatedObject`.

Figure 36 depicts a zoom on the KPI Module, with the addition of Time Ontology's classes and `KPISet` classes. Figure 37 illustrates an example graph based on this module.

Figure 34. Example graph based on the Equipment & Sensor module. HVAC system network serving a building zone.

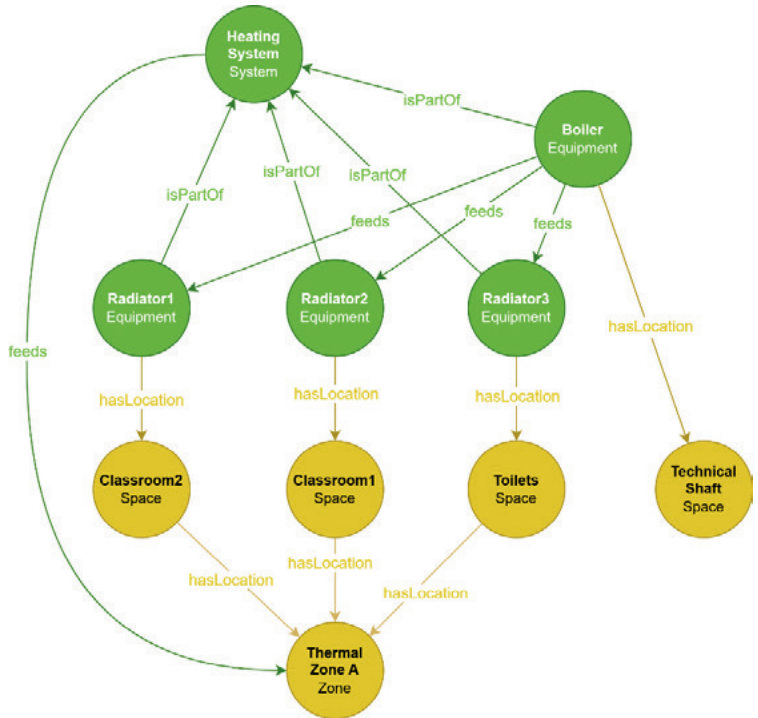
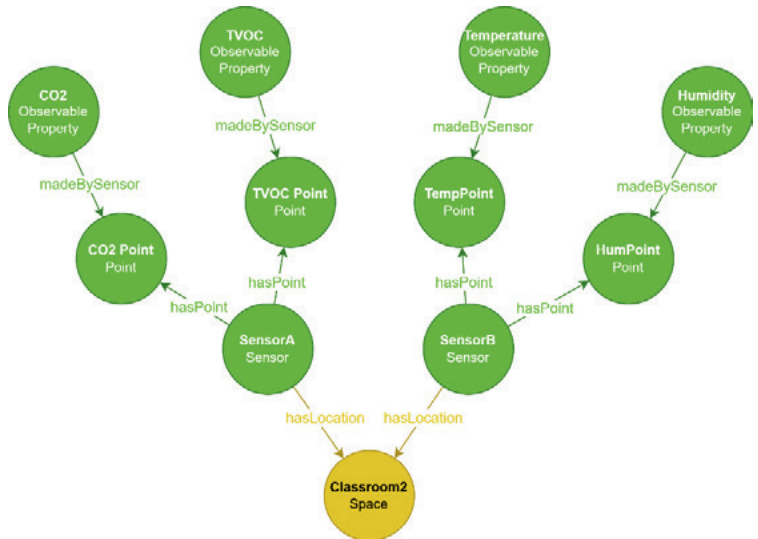


Figure 35. Example graph based on the Equipment & Sensor module. Sensing network serving a space.



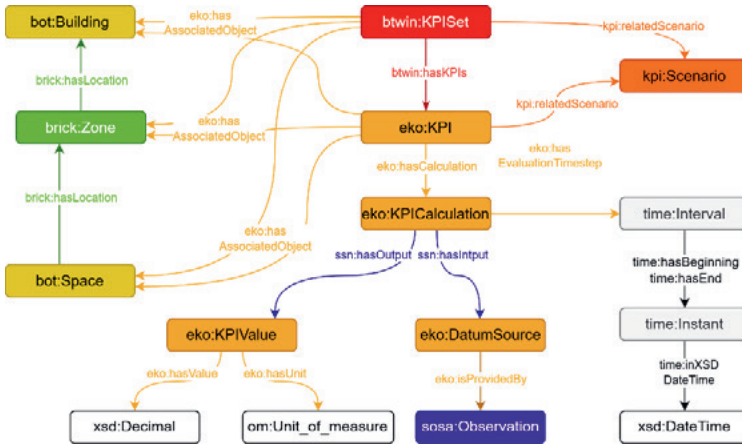


Figure 36. The KPI Module with the addition of time and kpi sets classes.

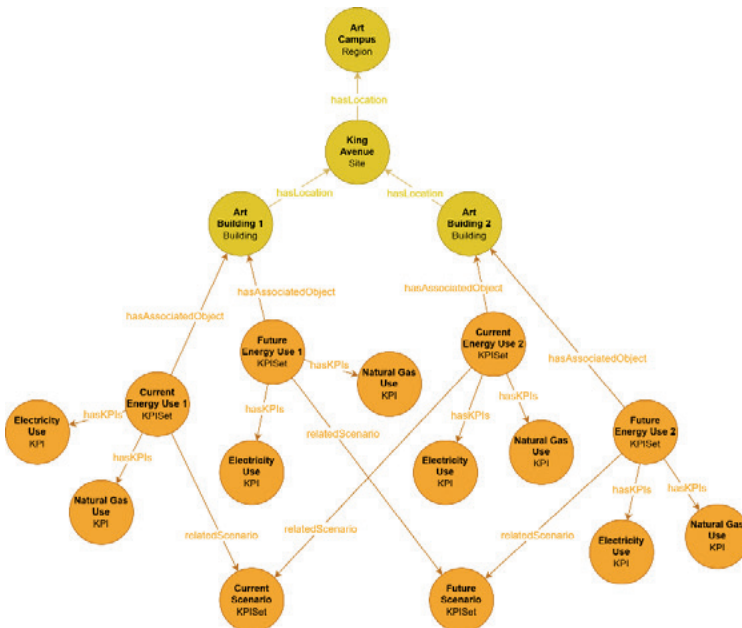


Figure 37. Example graph based on the KPI module. Energy management KPIs for a university campus.

3.3.7 Document Module

The *Document Module* is a simple module that was added to the FOF to link external documents, models, or data files to the graph entities they pertain to.

Here, the core class `btwin:Document` was introduced to represent an external resource (which could be a BIM file, a simulation model input, a PDF report, an image, etc.), and the `btwin:hasDocument` relationship to attach document to spatial elements, like `bot:Building`, `brick:Zone`, and others, as well as `kpi:relatedScenario` to connect documents or files to the scenario they belong to¹⁷.

This approach enables maintaining traceability between data and sources, allowing a user of the KG to navigate from a KPI value back to the source data or model that produced it. It is a lightweight mechanism to ensure that even if not all data is ingested into the graph (for example, complete simulation files might be too large or not fully translated), there is at least a pointer to them.

An example graph relying on the Document Module is provided in Figure 38.

3.3.8 Property Module

The *Property Module* allows mapping arrays of descriptive properties to the core structural and system information.

The FOF inherits the ‘property set’ mechanism of IFC. In particular, the Property Module incorporates ifcOWL’s classes `ifc:PropertySet`, `ifc:Property`, and the linking relations `ifc:HasPropertySets` (to attach a set to an element) and `ifc:HasProperties` (to link properties within a set)¹⁸ (see Figure 39 for an example).

The discussion continues in the next chapter with the introduction of the BTwin toolkit, a KG operationalising toolkit built directly upon the FOF.

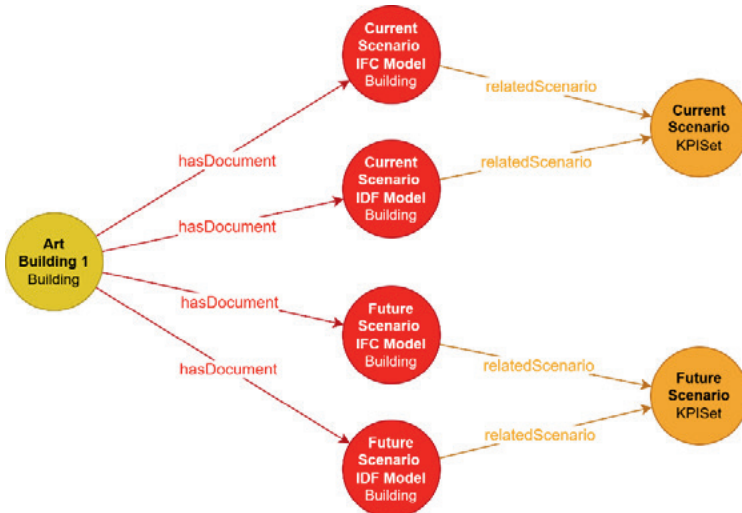


Figure 38. Example graph based on the Document Module. IFC and IDF files attached to a building, along with relevant scenarios.

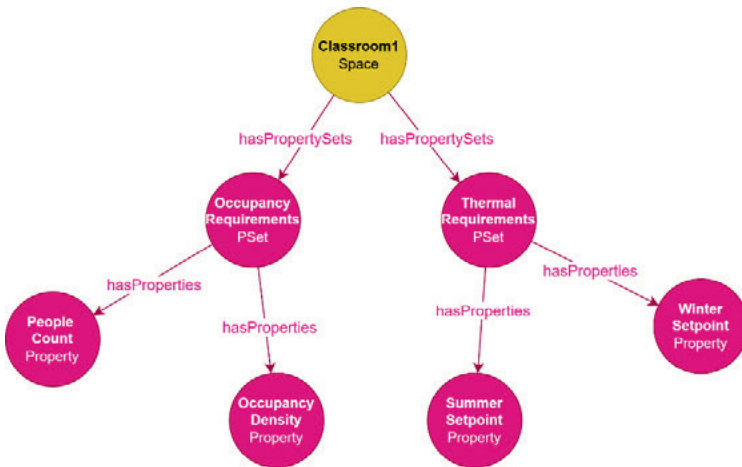


Figure 39. Example graph based on the Property Module. Occupancy and thermal requirements attached to a space.

Notes

¹ Nor is it likely that one ever will exist, since this would imply that humans are capable of representing the built environment in a single, definitive way, likely flattening its multiple forms and the diversity arising from local and contextual factors.

² This modular approach also reflects how expertise in building design and management is distributed across different disciplines – architecture, structural engineering, MEP, facility management, and others – each contributing its specific knowledge to shape a shared, systemic understanding of the building.

³ For instance, two ontologies might both have a concept for ‘Room’ but with different class hierarchies; the federation must either choose one as canonical or declare them equivalent via OWL axioms.

⁴ Although introducing a new bridge ontology is not ideal for standardising the process and may introduce vulnerabilities, it is a practical way to ensure the ontologies collectively form a connected graph of concepts rather than isolated islands.

⁵ For further exploration, reference can be made to the IFC documentation (buildingSMART International 2025).

⁶ EXPRESS is a standard for data modelling for product data. EXPRESS is formalised in the ISO Standard for the Exchange of Product model STEP (ISO 10303), and standardised as ISO 10303-11.

⁷ For instance, Brick included a ‘Location’ module (now deprecated) even though it was primarily concerned with building services.

⁸ These are: `bot:adjacentElement`, `bot:adjacentZone`, `bot:containsElement`, `bot:containsZone`, `bot:interfaceOf`, `bot:intersectingElement`, `bot:intersectsZone`, `bot:has3DModel`, `bot:hasBuilding`, `bot:hasElement`, `bot:hasSubElement`, `bot:hasSpace`, `bot:hasStorey`, and `bot:hasZeroPoint`.

⁹ Examples include: domain ontologies for building elements and distribution elements; ontologies for damage monitoring of buildings and built structures; ontologies for bridges; flow system ontologies; ontologies for building products; ontologies for geometry formats; ontologies for managing properties; and ontologies for managing geometry. Moreover, extension ontologies such as QUDT, SSN/SOSA, O&M and time can be combined with BOT, enabling adaptation to specific needs. (Lygerakis et al. 2022).

¹⁰ Accordingly, Brick can support queries such as ‘What types of sensors are installed in Room 101?’, whereas SSN/SOSA enables questions like ‘What was the temperature in Room 101 at 10 a.m. on 11th November 2025?’ by linking the observation to the space through the sensor’s location.

¹¹ The SMART framework explains this idea well (Doran 1981). It provides a structured approach to summarising data into meaningful indicators, outlining essential characteristics that KPIs should exhibit: Specificity, Measurability, Achievability, Realism, and Time-boundedness.

¹² A `brick:Zone` can represent, for example, a thermal zone (`brick:Energy_Zone`) or a fire compartment that spans multiple rooms (`brick:Fire_Zone`), while `brick:Portfolio` might group buildings under a single owner.

¹³ By including these simple concepts, KGs built on the FOF can support spatial reasoning about adjacency and connectivity of spaces; for instance, via these connections, it is possible to traverse from one space to another algo-

rhythmically, which is the basis for analysing evacuation routes, zones of influence in the layout, or containment relationship.

¹⁴ As we will see in the following chapters, to optimise computational performance, particularly when dealing with large volumes of sensor data, the observation records will be stored directly in the KG. Instead, the Equipment & Sensor Module will enable the KG to represent which equipment is present in a building, where it is located, how it is interconnected, and what data it produces, that is, the sensing topological network. Single measurements will instead be stored in alternative time-series databases.

¹⁵ In the context of the research, a scenario might be intended both as a particular past time period (e.g. 'Winter 2024 Heating Season') or a future condition state to evaluate with KPIs (e.g. 'Renovation with added insulation').

¹⁶ For example, a **KPISet** may refer to all indicators related to energy efficiency under a certain scenario, another to IAQ evaluation under another scenario, etc.

¹⁷ For instance, an EnergyPlus simulation input file (IDF) could be linked as a **btwin:Document** to a particular scenario node, indicating that file was used to generate the KPIs for that scenario.

¹⁸ For example, if a BIM model has an 'OccupancyRequirements' property set attached to a space with a property 'OccupancyNumberPeak', in the graph that would appear as an instance of **ifc:PropertySet** connected to the space node, with a child **ifc:Property** for occupancy number peak.

4. OPERATIONALISING GRAPH-BASED WORKFLOWS

Abstract

This chapter introduces the BTwin toolkit, a software framework designed to support the construction of Knowledge Graphs (KGs) for performance-based management.

Based on the Federated Ontology Framework (FOF) presented in Section 3.3, the toolkit provides a low-code approach to enable the operationalisation of graph-based workflows for prototyping digital microservices in building management.

In particular, it provides a reusable and extensible software library that, thanks to its modular design, facilitates data extraction from models and databases, semantic graph construction, integration of static information and dynamic time series, as well as the processing and visualisation of KPIs. The tool serves as the instrumental foundation for the demonstrative applications presented in the final chapters of the book.

In the text, the toolkit is explained in detail, starting with a motivation for its need, proceeding to a description of its architecture, the supported data serialisation format, and the main modules and functions. Finally, fictional usage examples are provided to anticipate the chapters that follow.

4.1 The need for a toolkit

As seen in the previous chapters, the fragmentation of building information is one of the most recurrent problems in digital Facility Management (FM). Decision makers confront a heterogeneous landscape of data repositories, which makes it challenging to obtain a unified, systemic view of buildings' performance.

Recent research has proposed KG-based approaches to provide answers to the data integration gap. However, the practical implementation of such knowledge systems still requires significant programming expertise, domain competence, and a vast amount of work. Added to the absence of standardised definitions, the heterogeneity of data sources, and the unsystematic approach that characterise built asset management (Abuimara *et al.* 2021a). This aspect hampers the streamlined development of such technologies.

As a consequence, developing these systems as digital products remains resource-intensive, with economic, financial, and cultural benefits that can sometimes be challenging for stakeholders to understand at the conceptual stage.

To accelerate the development of graph-based knowledge systems for building performance and make them resource-effective, the BTwin toolkit was developed. It is a software library that provides functions enabling the semantic integration of building data from heterogeneous sources within graph networks – including Building Information Models (BIM), building performance simulation (BPS), meters, and sensors – along with methods to utilise these knowledge bases for building interactive visualisations.

BTwin organises data into graphs that adhere to the axioms and semantics defined by the FOF introduced in the previous chapter. The FOF serves to align the semantic representation of data coming from fragmented sources, ensuring their consistency with KG standards. The resulting KGs integrate entities distributed across these data sources within a Federated Data Modelling (FDM) environment, creating a coherent knowledge base that supports the development of data web applications, through which non-expert users can query the KG via a Graphical User Interface (GUI). The aims of the toolkit are threefold:

- to unify static and dynamic building performance infor-

- to offer a set of high-level functionality functions to support rapid prototyping of *web applications* based on Business Intelligence (BI) principles;
- to promote an open and *collaborative development model*; the toolkit is distributed under a permissive licence and leverages existing opensource projects for BIM parsing and graph management.

A central design principle of the BTwin toolkit is its use case-agnostic nature. Rather than hard-coding specific performance indicators or domain models, the toolkit provides a generic method for digitally representing spatial objects, building systems, equipment, measurement points, and performance metrics. In this way, developers can utilise the BTwin framework to manage data, which can be generated through their own customised pipelines for various purposes. This flexibility allows the toolkit to be reused across projects of varying scales and objectives, as demonstrated later.

4.2 Toolkit architecture

4.2.1 Architecture overview

BTwin is implemented as a Python library¹. Semantically, the software architecture closely adheres to the FOF framework. Specifically, the following software specifications enable the operationalisation of the FOF and their serialisation into electronic formats:

- Hybrid data architecture for managing both static and dynamic data;
- Use of graph database engines as data storage layers;
- Implementation of connector logic to ensure semantic alignment with external schemas;
- Integration with openBIM standards and open-source software packages, from which the tool derives substantial benefits.

Figure 40 provides a high-level tech stack overview. Details regarding the system architecture are provided in the following paragraphs.

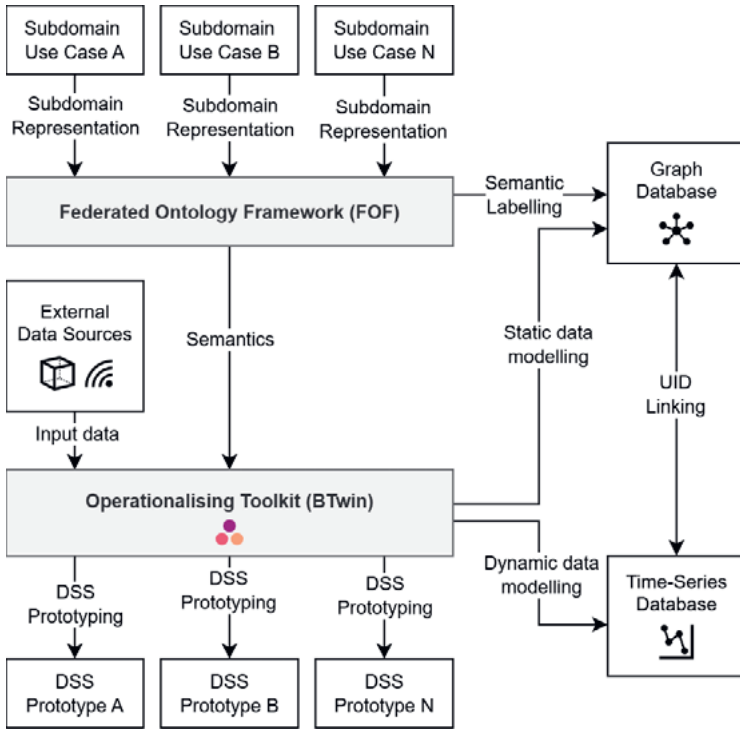


Figure 40. Overview of the tech-stack, centred on the ontology federation process coupled with an operationalising toolkit for DSS prototyping, supported by a hybrid architecture to manage static and dynamic data.

4.2.2 Hybrid data architecture

BTwin adopts a hybrid architecture that separately manages static and pseudo-static information from time-series data.

Static and pseudo-static data comprise information that changes little or not at all over time, such as data concerning building geometry, spatial topology, interface elements, building systems, and equipment networks. These datasets are highly interrelated according to a network logic and, in BTwin, are therefore represented as semantic graphs, in which nodes correspond to entities and edges encode relationships, as exemplified in the description of the FOF.

Dynamic data, by contrast, consist of time-series information, which, by their nature, change frequently and are updated in real or near-real time. Examples include data acquired from sensors (e.g., temperature, humidity, CO₂ levels), meters (e.g., power consumption, electricity usage), and simulations (e.g., thermal loads, solar gains, and similar variables). Storing such

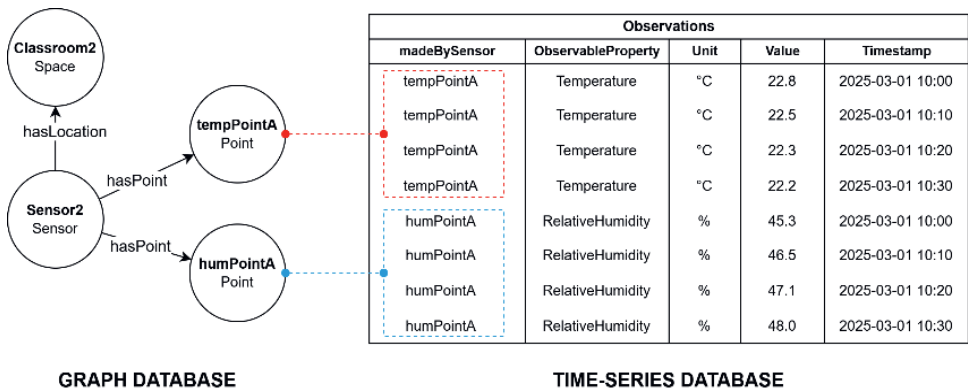
data directly within graph databases is inefficient for large time series, as each observation would correspond to a graph node, resulting in graphs containing millions of elements. To prevent computational bottlenecks, BTwin employs lightweight SQL databases to manage dynamic data. Static and dynamic data are linked through unique identifiers shared between the graph and SQL databases, ensuring that each observation stored in SQL can be traced back to its corresponding node within the graph database (Figure 41).

This approach enables the efficient storage and retrieval of extensive time-series datasets while preserving semantic context. In this way, graph operations remain computationally efficient, and time-series analyses do not compromise the integrity of the semantic model.

4.2.3 Database engines

Depending on the application's scale and purpose, BTwin adopts different back-end engines to manage graph data. For lightweight prototypes, BTwin employs NetworkX, a Python library designed for creating, manipulating, and analysing complex networks (NetworkX, 2025). While NetworkX is not a true graph database engine (for instance, it does not include a query language), it is simple to use and highly efficient for working with small or medium-sized graphs. In addition, it provides a wide range of built-in tools for network analysis, such as shortest path computation, minimum spanning trees, and centrality measures, making it particularly helpful for network analysis.

Figure 41. Connection between the graph database and the time-series database through unique identifiers.



For persistent storage and large-scale applications, BTwin integrates with Neo4j, an open-source, high-performance graph database (Neo4j 2025a). Neo4j uses a native graph storage format, optimised for efficiently storing and querying highly connected data. Unlike NetworkX, Neo4j provides a dedicated query language, called Cypher, which supports powerful graph traversal and pattern-matching capabilities. This makes Neo4j suitable for large datasets containing millions of nodes and edges, ensuring scalability and robust query performance.

In addition to graph databases, BTwin uses relational databases to manage dynamic and tabular data. In particular, it leverages SQLite, a lightweight, full-featured SQL database engine implemented as a C library (SQLite Consortium 2025). SQLite is beneficial during prototyping because it requires no separate server process and can be embedded directly into Python applications, providing reliable and efficient data management while maintaining a minimal setup overhead.

4.2.4 Connector logic

To enable data integration, BTwin relies on a set of connectors that extract information from external, fragmented sources, align it with the FOF's semantic schema, and perform graph serialisation.

Each connector is built upon an external library on which BTwin depends, providing the methods required to process specific file formats or data types.

For instance, the *IFC connector*, built on the *IfcOpenShell* library (IfcOpenShell n.d.), enables the traversal of an IFC file to identify spaces, building elements, and property sets, and to create the corresponding nodes and edges in the graph. It also computes basic topological relationships such as spatial containment hierarchies. The *EnergyPlus connector*, on the other hand, leverages the *Eppy* library to read EnergyPlus Input Data Format (IDF) files (Philip 2020). It extracts information from thermal zones, HVAC system definitions, schedules, and simulation parameters, and links the CSV output files containing the simulation results to the KPI nodes in the graphs. Another connector, based on the *Topologicpy* library (Jabi 2024), allows BTwin to compute topological relationships between spatial objects and boundary surfaces, facilitating the construction of

adjacency and navigation graphs.

Additional connectors interface with sensor data sources, reading time-series data from SQL databases and processing it to calculate KPIs. Other connectors are designed for data visualisation and dashboard development.

The complete list of BTwin's key dependencies includes:

- *IfcOpenShell* (IfcOpenShell n.d.): an opensource IFC toolkit that provides Python bindings for reading, writing and modifying IFC files.
- *Topologicpy* (Jabi 2024): a software development kit and plug-in that enables logical, hierarchical and topological representation of spaces and entities, facilitating the computation of adjacency and connectivity networks.
- *Eppy* (Philip 2020): a Python library for managing EnergyPlus input and output files. It enables programmatic navigation, search, and modification of EnergyPlus IDF files, as well as the extraction of simulation results.
- *NetworkX* (NetworkX developers 2025): a Python package for the creation, manipulation and study of networks. It provides data structures for graphs and standard graph algorithms, as well as methods for calculating graph measures.
- *Neo4j* (Neo4j Inc. 2025): a graph database management system that efficiently stores and queries highly connected data. It utilises native graph structures, offering high performance and scalability.
- *SQLite* (SQLite Consortium 2025): a self-contained, fast and reliable SQL database engine implemented as a C-language library. It is used for managing timeseries and other relational data.
- *Plotly* and *Dash* (Plotly 2025): interactive plotting libraries for Python, usable to create web-based dashboards.

Other minor dependencies include *Pandas* (pandas 2025) for data analysis and *NumPy* (NumPy Team 2025) for tabular data manipulation.

Figure 42 illustrates the BTwin ecosystem.

4.2.5 OpenBIM and open source principles

Connecting BTwin with well-established open-source libraries reinforces its alignment with the principles of openBIM².

The toolkit aims to embody these values by ingesting neutral, non-proprietary file formats to extract data for graph construction, by supporting open standards, and by being open source itself. This open-source nature allows users to inspect the source code, extend existing modules, and contribute to ongoing development, encouraging peer review, fostering transparency, and promoting integration within the academic research community.

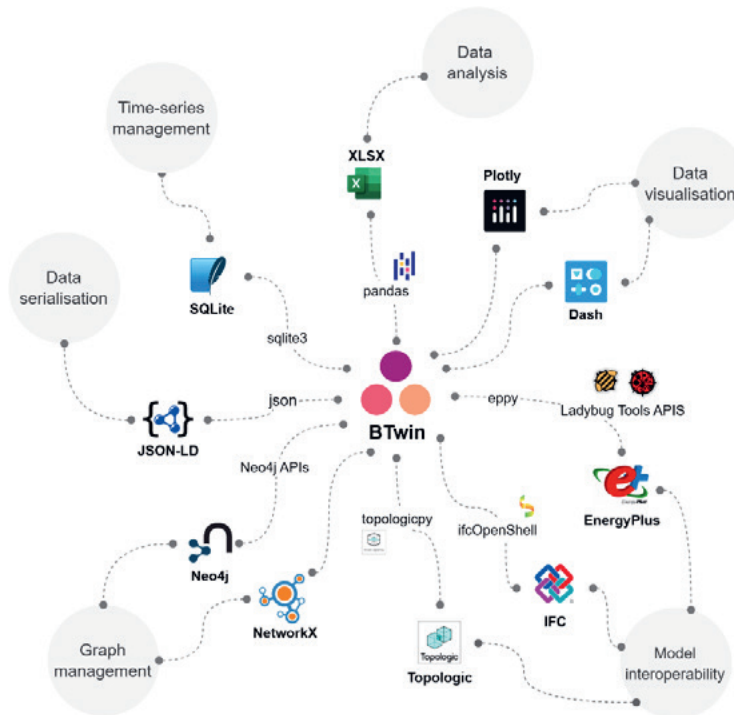


Figure 42. BTwin's connector logic, dependencies and supported formats.

4.3 Data serialisation

4.3.1 Graph data serialisation

In BTwin, graph serialisation – the process of transforming unstructured data into a storable, transferable, and machine-readable format consistent with graph representations – is implemented through the JavaScript Object Notation for Linked Data (JSON-LD)³. This notation is employed in the toolkit as an intermediate serialisation format for ingesting data from external sources into centralised KGs.

A typical JSON-LD document is organised into two main components (see Figure 43 for an example):

- the *@context* section, which defines mappings from compact terms used in the document to their corresponding full IRIs, ensuring unambiguous interpretation of keys and values, and referencing the ontologies that provide the vocabulary;
- the *@graph* section, which lists the graph's nodes as individual JSON objects, describing their metadata and class, and linking them through ontology-defined relationships to other entities.

In BTwin, this lightweight representation can be produced in two ways:

- *Manual object generation* can be performed using specific functions that create JSON-LD graph objects by specifying basic parameters, such as the element's unique identifier, class, and relationships.
- *Automatic object generation* can be realised through the connectors, which extract data from external sources and map it according to the JSON-LD syntax.

```

{
  "@context": {
    "bot": "https://w3id.org/bot#",
    "brick": "https://brickschema.org/schema/Brick#",
    "ifc": "https://standards.buildingsmart.org/.../OWL#",
    "name": "http://schema.org/name"
  },
  "@graph": [
    {
      "@id": "storeyX",
      "@type": "bot:Storey"
    },
    {
      "@id": "space1",
      "@type": "bot:Space",
      "relationships": {
        "brick:hasLocation": [
          {
            "@id": "storeyX",
            "@type": "bot:Storey"
          }
        ]
      }
    },
    {
      "@id": "sensor1",
      "@type": "ifc:Sensor",
      "relationships": {
        "brick:hasLocation": [
          {
            "@id": "space1",
            "@type": "bot:Space"
          }
        ]
      }
    },
    {
      "@id": "temperaturePoint1",
      "@type": "brick:Sensor",
      "relationships": {
        "brick:isPointOf": [
          {
            "@id": "sensor1",
            "@type": "brick:Sensor"
          }
        ]
      }
    }
  ]
}

```

Figure 43. Example of JSON-LD serialisation. The representation shows humidity and temperature measurement points linked to a sensor, <sensor1>, which is positioned in <space1>, located on <storeyX>.

4.3.2 Time-series data serialisation

A different serialisation strategy is used for handling time-series sensor data:

- For time series already available in structured data formats (e.g., EnergyPlus CSV output files), dedicated functions and connectors are developed in BTwin to access and query the data directly from the native time-series formats;
- For time series without a predefined structured format (e.g., raw sensor readings), data are stored in SQLite databases, following the template illustrated in Table 8. Example template for a SQL database containing sensors' data.. This tabular template includes the following columns: `sosa:madeBySensor` (the UID of the corresponding `brick:Point` in the graph), `sosa:ObservableProperty` (the measured quantity, e.g., temperature, humidity), `Unit`, `Value`, and `Timestamp`.

The template is organised so that, at query time, the system can retrieve the relevant time-series segment related to a graph object (like a sensor point) on demand. This hybrid approach ensures that the graph remains performant and focused on representing relationships and metadata, while detailed time-series data can be accessed when required. Conceptually, the FOF still defines the schema for time-series through the SSN/SOSA classes `sosa:madeBySensor` and `sosa:ObservableProperty`; however, instead of materialising every individual observation as a graph node, these instances are made available through SQL databases, with URIs of the graph nodes pointing to external data sources.

Table 8. Example template for a SQL database containing sensors' data.

<code>sosa:madeBySensor</code>	<code>sosa:ObservableProperty</code>	<code>Unit</code>	<code>Value</code>	<code>Timestamp</code>
<code>temperaturePoint1</code>	Temperature	°C	22.5	2025-03-01 10:00
<code>temperaturePoint1</code>	Temperature	°C	22.7	2025-03-01 10:10
<code>temperaturePoint1</code>	Temperature	°C	22.8	2025-03-01 10:20
<code>temperaturePoint1</code>	Temperature	°C	22.3	2025-03-01 10:30

4.4 Main modules and functions

4.4.1 Modular architecture

BTwin is organised as a collection of Python modules grouped into packages that correspond to major functional areas (Figure 44). The following overview summarises the main packages and their roles.

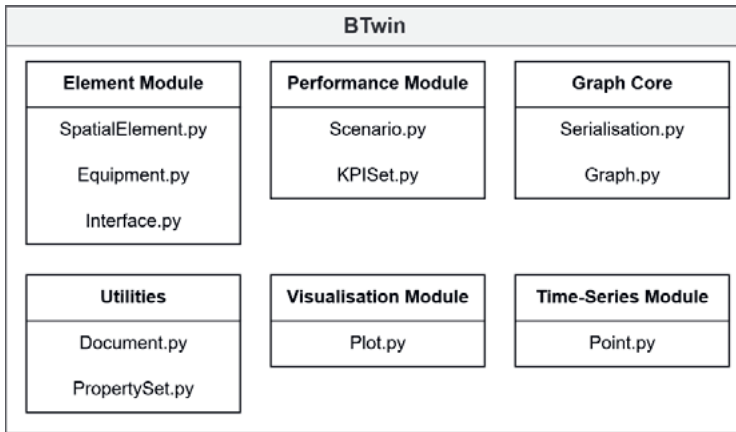


Figure 44. BTwin's modular architecture.

4.4.2 Element module

A first module contains classes and methods for extracting and mapping elements from native formats into graphs. In particular:

- The `SpatialElement.py` package handles the modelling of the FOF's spatial hierarchy, which can be executed manually or by automated extraction of the spatial hierarchies from IFC, IDF, or Topologic models.
- The `Equipment.py` package performs serialisation of equipment and sensing systems and connection to the spatial context.
- The `Interface.py` package provides methods for serialising interface elements, which can be done manually or by parsing IFC, IDF, or Topologic files.

4.4.3 Performance module

A second module allows for managing data concerning performance, like KPIs and scenarios.

- `KPISet.py` defines groups of performance indicators and associated calculation methods for specific tasks (e.g., comfort analysis, energy audits, etc.);
- The `Scenario.py` package allows for modelling hypothetical or predictive scenarios and managing scenario-specific data.

4.4.4 Graph core module

BTwin's core module offers methods for creating and managing the graphs. This core is composed of two packages:

- The `Serialisation.py` package gives functions to define, check, and correct the JSON-LD graph representation and validate its semantic content according to the FOF.
- The `Graph.py` package provides functions for creating actual graph databases in NetworkX or Neo4j. Static methods perform tasks such as JSON-LD to NetworkX or Neo4j conversion, node and edge creation, assignment of ontological types, graph merging, query execution, and graph export.

4.4.5 Timeseries management module

A third module comprises functions for manipulating time series data. It is composed of the `Point.py` package, which enables the management of dynamic data points such as sensor measurements or simulation outputs. This package defines specific methods for creating or handling sensor points and linking observations to graph nodes, as well as functions to create, update, and query measurement points, and to interface with SQLite databases.

4.4.6 Data visualisation module

The visualisation module relies on the `Plot.py` package, which provides functions for generating interactive plots, dashboards, and graph visualisations. It leverages *Plotly* and *Dash* to build web-based dashboards that enable users to explore performance data and indicators, compare scenarios, and drill down

into specific spatial entities or systems. Example visualisations include scatter plots of sensor values over time, heat maps of temperature distribution, bar charts of energy consumption per zone, and interactive diagrams of the graph.

4.4.7 Utilities

The `utils` package contains supporting functions that assist with naming conventions, validation and data consistency:

- `Document.py` manages attachments and metadata associated with reports or regulatory documents.
- `PropertySet.py` maps property sets and properties and assists in unit conversion and validation.

4.5 Usage examples

The third part of the book presents five demonstrative use cases based on the BTwin toolkit, which were developed in recent years as part of several research projects conducted in the Department of Architecture at the University of Bologna. The cases are organised in increasing semantic complexity to illustrate the versatility of graph-based workflows and are:

- Spatial graphs for the analysis of architectural layouts* (Chapter 5): BTwin is used in concert with `Topologicpy` and `NetworkX` to construct and analyse navigation graphs, i.e. graphs representing the connections between spaces through doors. Applied to a sample of one hundred residential floor plans, this application enabled typological and comparative analysis of the architectural layouts of some buildings built in Bologna after the Second World War. The focus of this demonstration was on the use of graphs for spatial analysis, with limited attention given to data integration and semantic enrichment issues. Time-series data were also excluded from this analysis.
- Topological graphs for safety evaluation of evacuation networks* (Chapter 6): BTwin is used in concert with `Topologicpy` and `NetworkX` to create topological graphs mapping the relationships between spaces and construction elements, allowing representation of evacuation routes in higher-e-

education buildings. This representation is particularly valuable for assessing the safety of the evacuation networks and supporting wayfinding applications. Even if more complicated than the first case, the use of graphs is still oriented towards spatial reasoning, with limited discussion on the topic of data integration and no reference to time-series management.

- iii) *Semantic graphs for data integration in heritage building characterisation processes* (Chapter 7): rather than on spatial reasoning, the third demonstration focuses on the use of graphs as data structures enabling semantic interoperability across diverse data sources. Here, BTwin is used to link BIM models with sensor and meter data, creating a knowledge base that feeds a prototype web application. Time series are managed here through the SQLite connector. This application enables building managers to access all relevant information regarding the performance characteristics of several historic museum buildings situated in an archaeological park.
- iv) *Performance graphs for decision support in energy management* (Chapter 8): in this demonstration, BTwin is used to create a federated KG integrating data from IFC and EnergyPlus models, thus allowing BIM-to-BEM interoperability. EnergyPlus simulations are run to produce fine-grained time series at the zone levels and calculate relevant KPIs for building energy management. These KPIs are stored in the KG, which is then queried to perform data visualisation through interactive plots. The primary focus of this use case is on investing in the interaction between occupancy and energy loads in a higher education building.
- v) *Scenario graphs for multi-criteria decision making* (Chapter 9): the last use case extends the temporal dimension of BTwin's graphs to encompass not only past data (e.g., historical time series) but also projected future scenarios. In this demonstration, the graph provides a knowledge base for supporting multi-criteria decision-making in occupancy management. Each scenario is represented as a sub-graph of a central federated KG that integrated BIM, energy, and safety models, along with energy- and safety-related KPIs. By comparing scenarios, decision-makers can as-

sess the effectiveness of potential occupancy management interventions on energy and safety, and identify optimal strategies. This is the most comprehensive use case, where graphs are utilised for both spatial reasoning (mapping evacuation systems in safety models) and data federation (linking BIM, BEM, and BSM data). Temporal data are also managed, not only in a monitoring sense but also in a predictive logic.

Notes

¹ Python is a programming language characterised by its straightforward syntax and widespread adoption within the software development community.

² OpenBIM is a collaborative approach to building design and management based on open, interoperable standards, which enables professionals from different disciplines, using diverse software tools, to exchange information seamlessly and transparently throughout a project's entire lifecycle, also thanks to standardised formats such as IFC.

³ JSON-LD is a lightweight syntax for expressing linked data using JSON. As a W3C recommendation, it was designed to ensure that data are interoperable, machine-readable, and semantically meaningful, while retaining ease of use for developers already familiar with JSON. At its core, JSON-LD embeds Linked Data principles (identifiers, contexts, and relationships) into standard JSON structures. This allows entities originating from different datasets to be disambiguated, connected, and integrated using Internationalised Resource Identifiers (IRIs). This format enables the graphs to be instantiated either in an RDF triple store or a Labelled Property Graph (LPG), depending on the use case requirements. JSON-LD is particularly valuable because it bridges the developer-friendly JSON syntax with the semantic web ecosystem, supports the serialisation of RDF data models, and facilitates interoperability across web applications, APIs, and KGs.

APPLICATIONS AND CASE STUDIES

5. SPATIAL GRAPHS FOR THE ANALYSIS OF ARCHITECTURAL LAYOUTS

Abstract

This chapter presents the first application discussed in the book, developed in accordance with the theoretical and methodological framework of *Buildings as Networks*.

This application focuses on the use of graph-based techniques for analysing building layouts. In particular, the text introduces a semi-automated workflow for generating graph representations of floor plans, applied to nearly 100 residential buildings located in Bologna. The resulting dataset supports typological analysis and comparison at the urban scale. Within this context, graphs prove particularly effective for examining the spatial structure of these buildings through graph analytics, with a primary focus on studying centrality and connectivity metrics. The emphasis of this chapter is on the use of ‘*graphs for topological analysis of networks*’ rather than ‘*graphs for data integration*’, an aspect addressed instead in later chapters.

The text documents the main outputs of the research presented in the article ‘*Semi-Automated Dataset Generation for Residential Buildings Using Graph-Based Topological Modelling*’ (Massafra *et al.* 2025b), aligning it within the book’s framework. Most of the work described here was conducted during a visiting period I had at Cardiff University (UK) in the early months of 2025, in collaboration with Wassim Jabi, Dania Al-Harasis, and Lorenzo Stefanini, to whom I am grateful for their contributions.

5.1 Contextualisation and use case definition

This chapter inaugurates the third part of the book, which presents case studies demonstrating how graph-based methodologies can be applied to existing building stocks for several purposes. While the previous chapters outlined the principles of representing Buildings as Networks and described the underlying semantics and ontological models, this and the subsequent chapters attempt to provide concrete examples that validate these concepts.

The use case presented in this section demonstrates how residential floor plans can be transformed into graphs representing architectural layouts and how, after being generated, these graphs can be automatically processed to infer the occupancy type of the spaces based on node connectivity, centrality, and morphological features.

The motivation behind using graphs for such analysis is that, as already mentioned, in architectural contexts, graphs can provide practical approaches to encoding both the connectivity of spaces and the hierarchy between them¹. By contrast, alternative modelling techniques, such as polygonal or voxel representations, can accurately describe space geometry – also better than graphs – but are less effective at capturing relational properties.

To operationalise the graph analysis, the presented research lays its foundation on Topologic's framework, which, as seen in Section 3.2.4, conceives buildings so that spaces can be identified as `Cell` elements, aggregated into a single building entity, the `CellComplex`, and interconnected through `Aperture` elements, representing doors. This conception enables the encoding of information into a `Topologic Graph` entity, which explicitly maps how spaces are connected through doors in the floor plan layout.

The application relies on a workflow capable of semi-automatically generating a dataset of residential layout graphs, starting from floor plans in raster formats, such as `.PDF`, `.JPG`, or `.PNG`. Each of these graphs consisted of nodes representing spaces and edges representing the passage between spaces through doors, with nodes storing information about space geometry and centrality.

The workflow consists of four fundamental steps: collecting raster planimetric drawings; retracing spaces and apertures in a CAD environment; generating topological graphs that map floor circulation; and assigning functional labels to spaces using conditional rules inferred from graph centrality metrics. The central aspect of the method involves the inference step, supported by an eleven-rule, conditional data modelling procedure that processes node attributes to detect the functional labels of nodes (e.g., entrance, corridor, living room, bedroom, kitchen, toilet, etc.).

The methodology is tested on a sample of Italian residential buildings constructed between the 1940s and 1960s. In addition to the methodology, the primary outcome of the research is a dataset comprising nearly 100 floor plan graphs, corresponding to approximately 400 apartments, produced in three different graph formats: Topologic's JSON, NetworkX's JSON, and BTwin's JSON-LD. The dataset serves as a typological analysis of this stock, offering an analytical perspective for characterising the distributive typologies of the buildings and revealing common design patterns in the residential architecture of the period.

5.2 Case study: post-war residential stock in Bologna

The case study for this application consists of public housing blocks built in Bologna (Italy) between 1945 and 1965 (Figure 45). These were chosen because they represent a significant portion of the Italian urban fabric. Indeed, nationwide, the residential stock is estimated to be around 30 million units, about 70% of which were built before the 1970s, before the implementation of modern seismic and energy regulations (Italian National Institute of Statistics (ISTAT) 2011).

Intrinsic construction and architectural features mark housing from the 1940s to the 1960s in the peninsula, like non-seismic conception of structural systems, low construction quality, uncertainty in the definition of material properties, and poor urban and architectural conformation of the areas where they are located, which make them particularly inadequate to the contemporary requirements. Addressing these shortcomings

often demands radical interventions that are costly, technically complex, or incompatible with heritage preservation.

Consequently, a key question confronting policymakers is whether to focus on preserving or renovating this heritage or opt instead for demolition/reconstruction strategies. Both approaches have socio-economic and environmental implications, necessitating a comprehensive knowledge of this heritage's physical, typological, and construction characteristics for informed decision-making.

If, on the one hand, it is recognised that such choices could be supported and enhanced by data-driven procedures – to be grounded in reliable knowledge bases – on the other hand, challenges exist in acquiring data at the urban level. Traditional urban mapping methods remain resource-intensive; moreover, existing digital databases (e.g., ISTAT or a municipality's GIS, as in the case of Bologna) often provide only general information about buildings, which is insufficient for targeting interventions at both urban and building scales. The research, therefore, underscores the need for systematic approaches to collect and integrate large-scale data on architectural features into data-driven tools, such as city digital twins (Lehtola *et al.* 2022), to inform policy decisions more effectively.



Figure 45. Google Street View images of a subsample of the buildings under investigation.

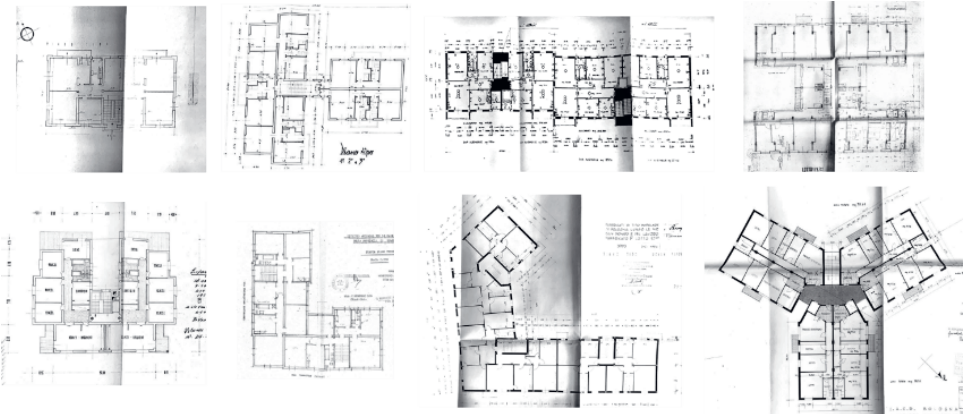


Figure 46. Raw floor plans for a sample of the investigated buildings.

5.3 Methods and tools

5.3.1 Workflow articulation

The methodology adopted to pursue the research objectives is organised into four main phases, described in the following sections:

1. Data acquisition;
2. Data preprocessing;
3. Data modelling;
4. Data processing and analysis.

5.3.2 Data acquisition

In the first phase, all the raw data required for the analyses were acquired. The workflow began with the collection of planimetric drawings of the buildings' typical floor plans. These consist of the primary input for the workflow. In the context of this study, the drawings were retrieved from the Historical Archives of the Emilia Romagna Region (ParER). They consisted of paper technical drawings of floor plans, sections, and elevations from the original design, along with related technical documents². The plans were scanned and made available as image raster files in PNG format. The selected sample of buildings comprised 98 multi-storey public houses built in Bologna between 1945 and 1965, corresponding to 175 blocks and 392 apartments. Figure 46 depicts a subsample of the scanned drawings.

5.3.3 Data preprocessing

Because the scanned images contained visual noise (such as annotations, symbols, and scale distortions), they could not be directly analysed without first being vectorised. In the second phase, preprocessing techniques were therefore applied to convert the raw drawings into digital models.

The preprocessing step proceeded as follows:

- The images were imported into Rhino, a 3D modelling environment that supports geometric manipulation (Robert McNeel & Associates 1993).
- In Rhino, the spaces of each floor plan were manually re-traced³ (see Figure 47).

- The space boundary curves were scaled to real-world dimensions.
- Using Rhino in connection to Grasshopper (GH) ('Grasshopper - algorithmic modeling for Rhino' 2025), a Visual Programming (VPL)⁴ environment, the curves were aligned to a 10 cm grid to correct minor inaccuracies.
- The 2D curves were extruded into 3D closed objects.
- Through another GH algorithm, the 3D objects representing the spaces were converted into Topologic Cells.
- The cells were merged into a CellComplex encoding spatial adjacency.
- Dummy Apertures (representing doors) were generated and projected onto the cells' faces.
- Passage relationships between spaces through doors were computed, producing a *passage graph*.
- The CellComplex was exported as a Topologic file.
- Always working in GH, the passage graph was exported in CSV format.

Figure 48 illustrates the main results of this step, where the visualisation of the BReps⁵ is superimposed on the visualisation of the passage graph.

Several challenges were encountered during the data preparation process. The manual retracing process proved to be the most critical step, as it was labour-intensive and prone to human error. In the early experiments, even minor discrepancies in tracing led to misaligned vertices or incorrect adjacency relationships. To address this, GH scripts were integrated into the workflow to ensure consistency in the retracting operations⁶.

Another difficulty arose from the heterogeneous quality of the archival drawings, as some plans were faded or incomplete, necessitating interpretation by the operator. Data variability was mitigated by relying on a single operator for the retracing process, who was familiar with both the buildings and the associated documentation. In cases of uncertainty, cross-checking was performed using the original technical project documents or, where necessary, Google Maps to verify the geometry and location of the buildings.

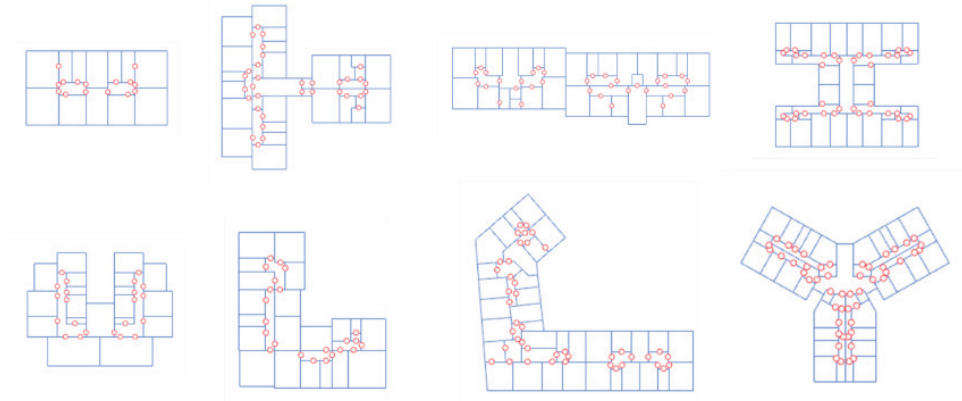


Figure 47. Retraced floor plans for a sample of the buildings under investigation.

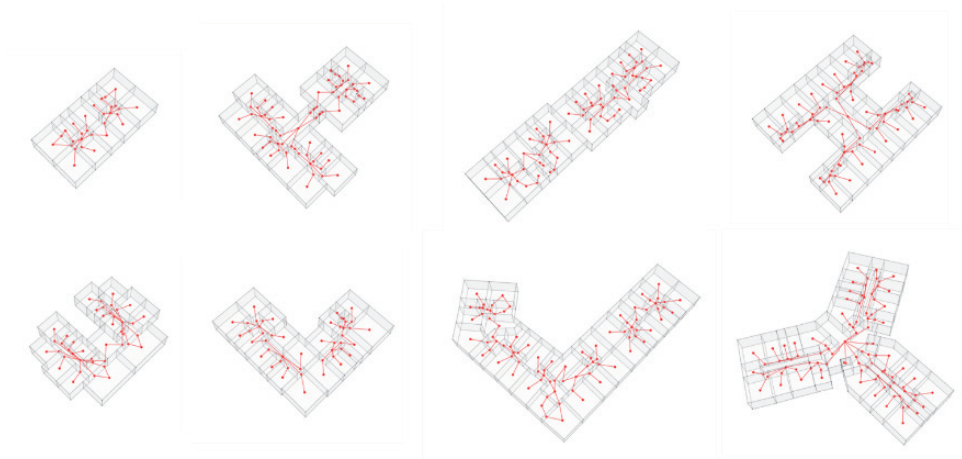


Figure 48. Topologic Graph and CellComplex for a sample of the buildings under investigation.

5.3.4 Data modelling

Once exported in CSV format, the graphs were imported into Python using Pandas (pandas 2025). Here, CSV files were processed to align with BTwin's JSON-LD notation and then converted into NetworkX graphs. Additional nodes and relationships were added to reconstruct the complete spatial hierarchy for each building.

All semantic attributions of the nodes and edges were aligned with the Federated Ontology Framework (FOF) presented in Section 3.3, particularly with a subset of concepts drawn from the Building Topology Ontology (BOT), Brick, and the Topologic Class Hierarchy, following the subpart of the ontology depicted in Figure 49.

Each node and edge in the graphs was labelled with its corresponding ontological class and assigned a unique identifier valid across the entire dataset.

The core ontological framework supporting this application was based on a simple spatial hierarchy, where:

- `bot:Building` represents each building.
- `bot:Storey` represents the storey in the building whose floor plan is analysed, i.e. the 'typical' floor.
- `brick:Zone` represents apartments, defined as aggregations of spaces.
- `bot:Space` represents each room within the floor plan.

These elements were related such that each `bot:Space` was connected to both `brick:Zone` and `bot:Storey` through the `brick:hasLocation` relationship. Similarly, location relationships were established between `bot:Storey` and `bot:Building`, as well as between `brick:Zone` and `bot:Building`.

In addition to the classes of the spatial hierarchy, interface elements were included. In particular, the `top:Aperture` class was employed to represent doors between spaces, linking two `bot:Space` entities and enabling passage. The `bot:interfaceOf` relationship linked `top:Aperture` to `bot:Space`, thereby creating the chain *space–aperture–space* used to map the passage graph of the floor. This was also expressed through the `btwin:hasPassageTo` relationship between two `bot:Space` entities.

To realise the graphs, a connector was developed to map Topologic elements within BTwin's NetworkX graphs. NetworkX was selected to manage graphs for this application, as it pro-

vides a lightweight yet powerful environment for graph management and analysis. On the one hand, it was chosen over alternatives like Neo4j, because it allows efficient handling of small datasets in a very manageable JSON format; on the other, because the NetworkX Python library includes a wide range of well-established functions handy for graph analytics, such as centrality metrics, minimum spanning tree, community detection, and so on.

In particular, the connector encodes:

- Topologic CellComplex as bot:Building.
- Topologic Cluster as brick:Zone.
- Topologic Cell as bot:Space.
- Topologic Aperture as top:Aperture.

Since Topologic does not natively include a class representing storeys, bot:Storey nodes were generated from the geometric data of the BRep file. Specifically, one storey was created for each unique z-coordinate corresponding to the lower surface of the BReps representing the spaces.

Figure 50 provides an example graph generated at this point.

In addition, to further prepare the graph, morphological and centrality metrics of the spaces were computed and attached to the nodes as properties. Morphological parameters were calculated using Topologicpy functions by inputting the BRep file (Table 9). Centrality metrics were instead computed through NetworkX (Table 10).

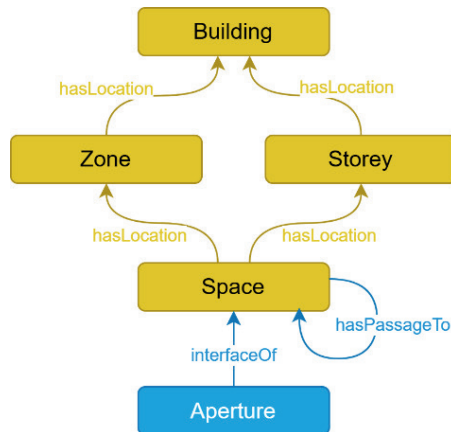


Figure 49. Subpart of the FOF used for the application.

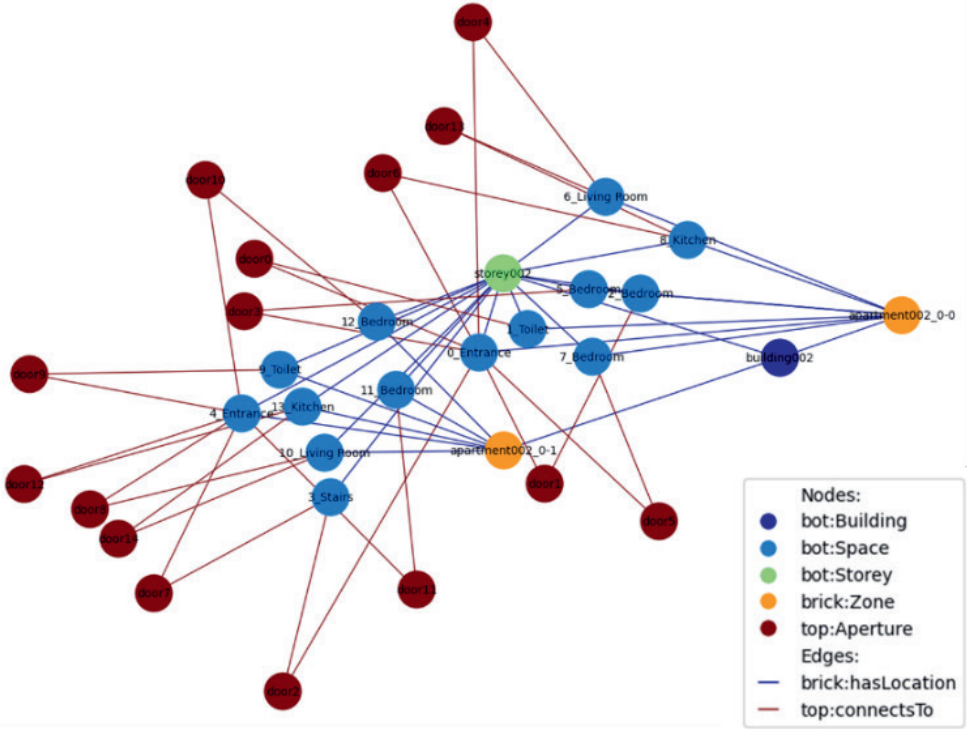


Figure 50. Example BTwin's graph representing a floor plan.

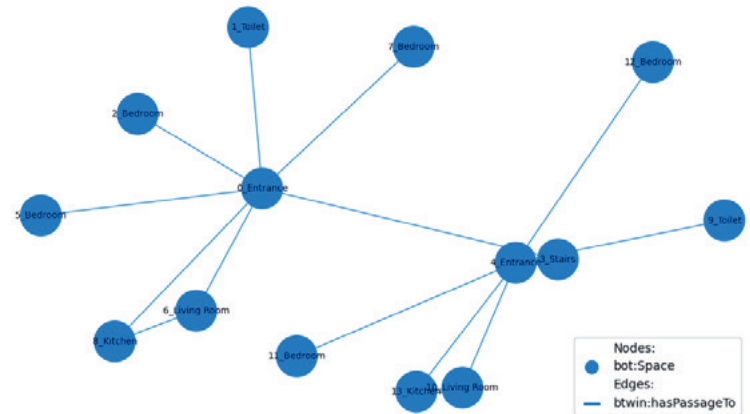


Figure 51. Example BTwin's graph representing a floor plan, simplified for the analysis.

Property Name	Description
Betweenness Centrality	The betweenness centrality value for the node in the graph. Betweenness centrality quantifies a node's significance within a network based on its position along the shortest paths between other node pairs, in which a node is considered central if it frequently serves as an intermediary in these shortest-path connections [122].
Closeness Centrality	Closeness centrality value for the node in the graph. Closeness Centrality suggests that a node's proximity to other nodes in the graph determines its closeness to the rest of the network [123].
Degree Centrality	Degree centrality value for the node in the graph. It is the total number of connections a space has relative to all other spaces within the architectural environment [124].
Eigen Vector Centrality	Eigenvector centrality for the node in the graph. Eigenvector Centrality evaluates a node's importance by accounting for both its direct connections and the significance of its neighbouring nodes [125].
Uniform Component Decomposition Centrality	Uniform component decomposition centrality of the node in the graph
Betweenness Centrality	The betweenness centrality value for the node in the graph. Betweenness centrality quantifies a node's significance within a network based on its position along the shortest paths between other node pairs, in which a node is considered central if it frequently serves as an intermediary in these shortest-path connections [122].

Table 10. Centrality metrics calculated through NetworkX.

Table 9. Morphological metrics calculated through Topologicpy.

Property Name	Description
Compactness	Measure of the space's compactness, defined as the ratio between the gross area of the space and the gross area of its smallest bounding rectangle in the xy plane.
GrossArea	Gross area of the property (in square meters).
Rectangularity	Measure of how closely the space's shape approximates a rectangle, defined as the ratio between the width and the length of the smallest bounding rectangle of the space in the xy plane.
SV Ratio	Surface-to-volume ratio of the property, defined as the ratio between the volume of the space and the sum of the areas of all faces bounding the space (walls, roofs, and floors).

Once ready, the refined graphs underwent an automated functional enrichment process. This process relies on a Python algorithm that, following an eleven-step procedure, allows inferring occupancy types of spaces from nodes' morphological and centrality measures (Figure 52).

These rule sets, described below, were determined after iterative and careful observation of the analysed floor plans, which, as mentioned, belong to a set of buildings with similar functions, ages, locations, and construction characteristics. These are:

- *Extraction of building blocks and identification of stairs* (Ruleset 1). Each floor plan graph is decomposed into connected subgraphs representing *building blocks*. Uniform Component Decomposition Centrality (UCDC) is introduced to identify the staircase node within each block⁷.
- *Detection of elevators* (Ruleset 2). Nodes with degree⁸ one connected to a staircase and with an area less than 50% of the staircase area are classified as elevators.
- *Extraction of apartment subgraphs and entrances* (Ruleset 3). Removing stairs and elevators yields subgraphs corresponding to individual apartments. Nodes directly connected to the staircase in the original block graph are designated as apartment entrances.

- *Corridor assignment* (Ruleset 4). Within each apartment graph, corridors are identified by computing the minimum spanning tree⁹ and marking nodes with a degree greater than one. These nodes, due to their connectivity, act as circulation spaces.
- *Differentiation between corridors and passthrough living rooms* (Ruleset 5). The previous corridor assignment rule sometimes misclassifies some circulation spaces as corridors. While these spaces function as circulation areas, they also serve as living rooms. To identify these ‘*pass-through living rooms*’, three node-chain configurations have been developed to reclassify nodes based on their connectivity characteristics: ‘*Entrance-Corridor-Corridor*’, ‘*Corridor-Corridor-None*’, and ‘*Entrance-Corridor-None*’. Reclassification occurs when the central corridor has a larger area and specific degree conditions relative to adjacent nodes.
- *Distinguishing liveable and supporting spaces* (Ruleset 6). A KMeans clustering¹⁰ procedure partitions unclassified nodes into ‘liveable’ and ‘supporting’ spaces, using both geometric and centrality attributes. Moreover, due to the sensitivity issues of K-Means – particularly misclassifying small bedrooms as supporting spaces – this method is enhanced by a simple rule that, in cases of inconsistencies, reclassifies all nodes with a gross area greater than 10 m² as living spaces.
- *Identification of living rooms* (Ruleset 7), *bedrooms* (Ruleset 8), *kitchens* (Ruleset 9), *toilets* (Ruleset 10), and *closets* (Ruleset 11). Subsequent rules sequentially label the remaining nodes. The living room is identified among the living spaces based on proximity to the entrance, degree (connectivity) and area; remaining living spaces are classified as bedrooms. Closets are identified as those spaces whose area is less than 30% of the average room area; kitchens are identified among the spaces connected to the living room based on size and adjacency; and all remaining supporting spaces are designated as toilets.

After the data enrichment process, all the graphs were saved in a JSON format compatible with NetworkX, with each node storing, in addition to its unique identifier, semantic label, centrality, and morphological metrics, its functional attribution.

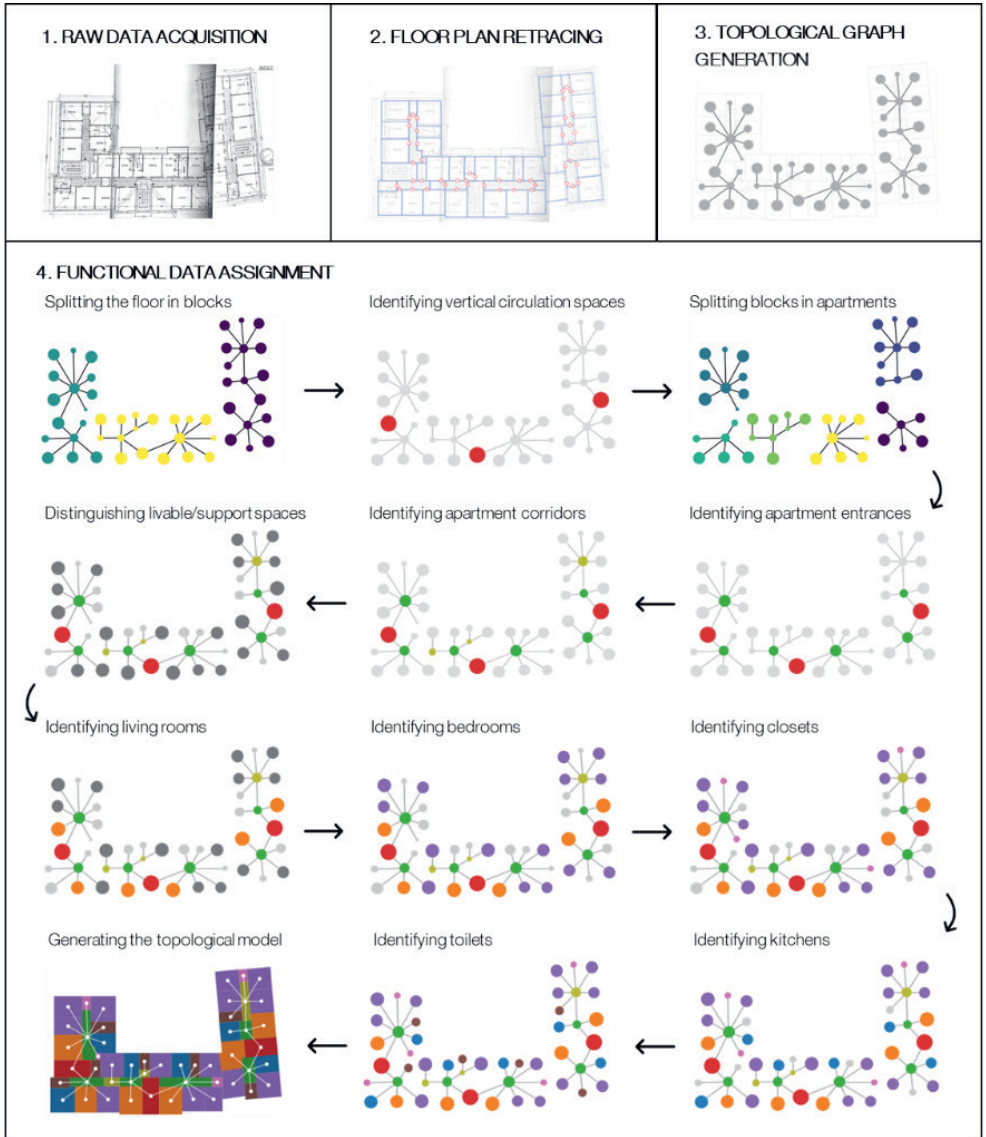


Figure 52. Main steps to infer node occupancy type using graphs.

5.4 Results and outcomes

5.4.1 Results overview

The workflow was applied to the 98 buildings (175 blocks and 392 apartments) that constitute the study sample, resulting in a consistent dataset of graphs suitable for comparative analysis (Figure 53 and Figure 54). Three main activities were carried out on this dataset:

1. Evaluation of the accuracy of the results achieved in the node classification process;
2. Statistical analysis of the study sample;
3. Sensitivity analysis of the most relevant metrics for predicting occupancy type.

5.4.2 Evaluation of process accuracy

Once the node classification methodology was applied to all 98 buildings, correction of misclassifications across the entire dataset was performed manually, both to make the dataset more consistent and to evaluate the process accuracy. As a result, approximately 10% of the nodes required adjustment, resulting in a 90% accuracy of the process. Most errors occurred between single bedrooms and kitchens, or between large bedrooms and living rooms, due to their similar areas and topological positions. The results demonstrate high potential in this context, where limited data were available, and alternatives such as ANNs and GNNs cannot be applied to the node classification task due to the limited sample size.

5.4.3 Statistical analysis of the dataset

After being checked and validated by a human operator, the graph dataset was analysed to understand the sample composition and characteristics. Figure 55, Figure 56, and Figure 57 illustrates the distribution of floor area and number of nodes at the floor, block, and apartment levels Figure 58 and Figure 59 depict instead the distribution of space occupancy types in the dataset.

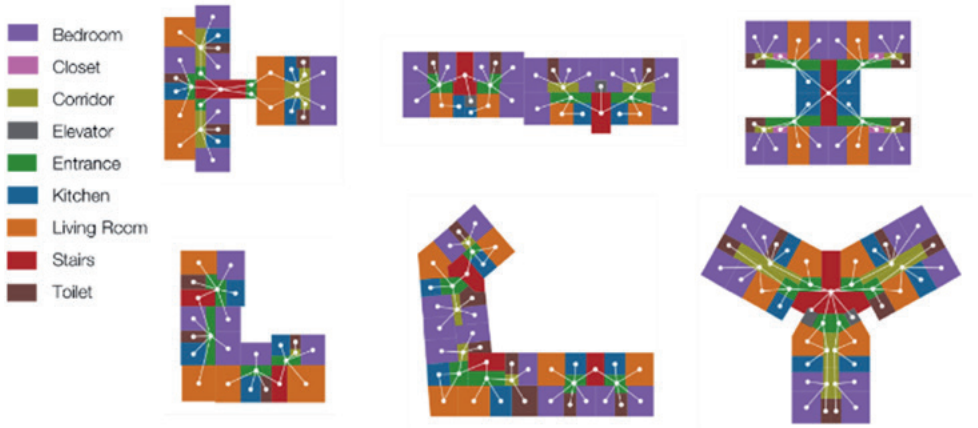


Figure 53. Topologic graphs (with cells coloured by space occupancy type) for a sample of the buildings under investigation.

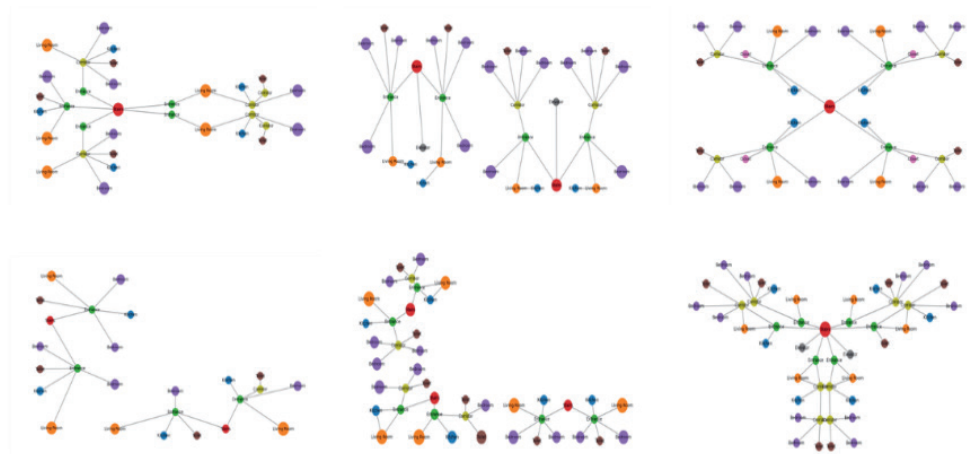


Figure 54. NetworkX graphs (with nodes coloured by space occupancy type) for a sample of the buildings under investigation.

The distribution of floor areas shows a right-skewed pattern, with the majority of floor plans being relatively small and only a few exceeding 1000 m². In terms of block configuration, a single block per floor plan is the most common arrangement, while multi-block layouts are less frequent. The number of apartments per floor plan is concentrated around the lower values, with a clear peak between 3 and 5 apartments per floor plan. Similarly, the distribution of nodes (representing spaces within the floor plan) also follows a right-skewed trend, suggesting a general preference for simpler designs with fewer internal subdivisions.

At the block level, most block areas fall within the range of 150–300 m². Blocks containing three apartments are the most frequent, while those with five are the least common, indicating a preference for three-unit configurations. The distribution of spaces within each block is slightly skewed, with the majority containing between 10 and 25 spaces. At the apartment level, apartment areas range from 50 to 160 m², with most falling within the 90–110 m² range. The number of nodes, corresponding to the number of spaces per apartment, varies between 2 and 30, with 7 being the most frequent value.

Bedrooms are the most numerous type of space and account for the largest share of total area. Living rooms also represent a significant proportion and are present in almost every apartment. Kitchens and toilets generally occur once per apartment. Most bedrooms have a gross area between 10 and 25 m², with a peak of around 18 m². Living rooms range from 14 to 29 m², with a peak size of approximately 21 m².

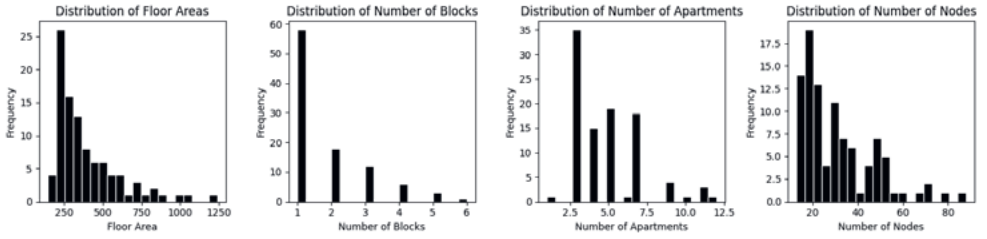


Figure 55. Distribution of floor plans by total area, number of building blocks, apartments, and spaces.

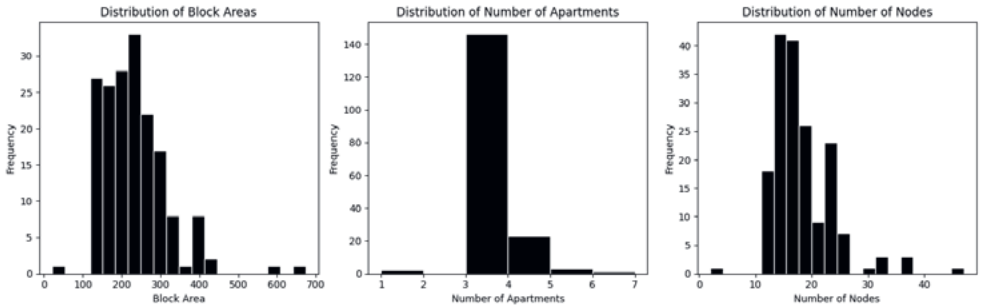


Figure 56. Distribution of blocks by total area, apartment count, and number of spaces.

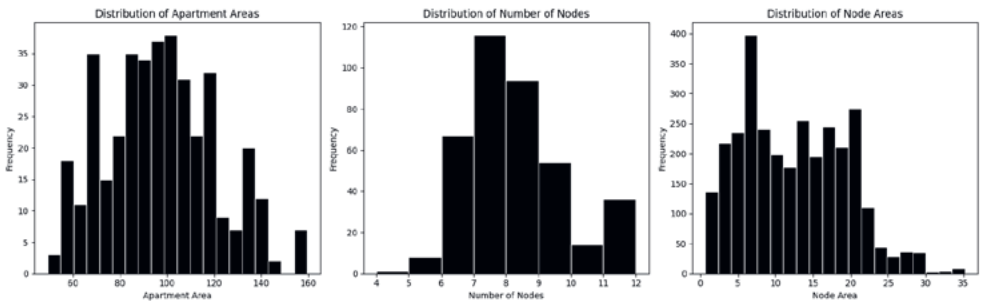


Figure 57. Distribution of apartments by total area, number of spaces, and number of edges.

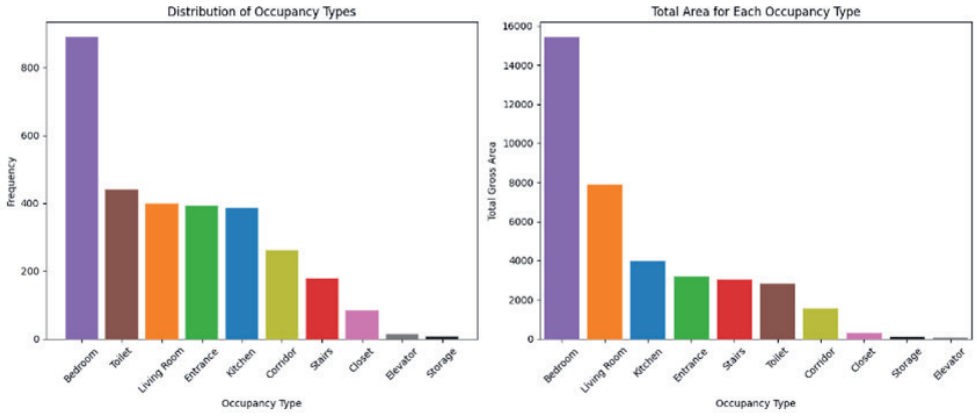


Figure 58. Distribution of functions in the dataset (by total count and area).

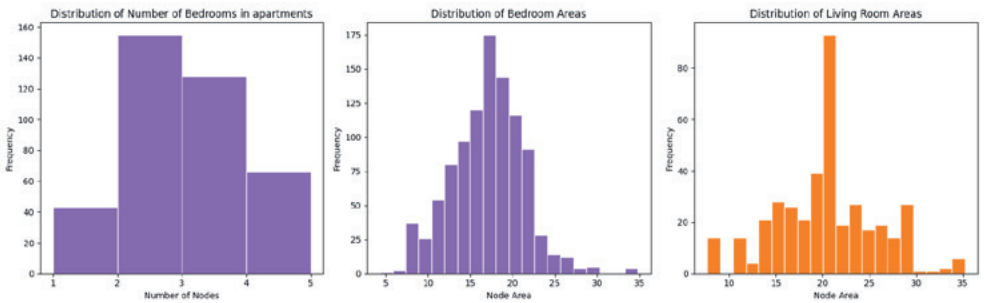


Figure 59. Distribution of bedrooms and living room (by total count and area).

5.4.4 Sensitivity analysis

A sensitivity analysis was conducted at the end of the process to determine which features were most influential in predicting the occupancy type of nodes. Initial tests with a Random Forest (RF) classifier¹¹, shown in Figure 60, revealed that shape and size metrics, particularly the surface-to-volume ratio and gross area, were the most influential. Topological descriptors such as closeness and betweenness centrality at the floor level also proved significant.

To further validate this finding, several Machine Learning (ML) models were trained on the dataset, including RF, XGBoost¹², and Artificial Neural Network (ANN)¹³, to predict occupancy type from both morphological and topological features. Each of these models was optimised through an iterative approach by tuning the primary hyperparameters. Three experimental configurations were then tested for each of them:

- a) Classification of node occupancy type by using only morphological descriptors;
- b) Classification of node occupancy type by using only topological descriptors;
- c) Classification of node occupancy type by using both morphological and topological descriptors.

XGBoost proved to be the best-performing algorithm when trained with the combined feature set (configuration C), achieving an accuracy in node classification of approximately 86%. By contrast, the models trained exclusively on morphological or centrality metrics achieved accuracies of only 61% (configuration A) and 63% (configuration B). The analysis of the confusion matrix¹⁴ revealed that most classification errors occurred between bedrooms and living rooms, or between smaller bedrooms and kitchens. These errors reflect the similarity between these spaces in terms of size and topological position, which makes them particularly difficult to distinguish.

Overall, the results highlight the methodological importance of integrating morphological descriptors and graph-derived metrics within a unified analysis framework. Graph representations made it possible to capture both geometric properties and circulation patterns simultaneously, thereby enriching the feature space available to ML and enabling reliable functional classification of residential layouts.

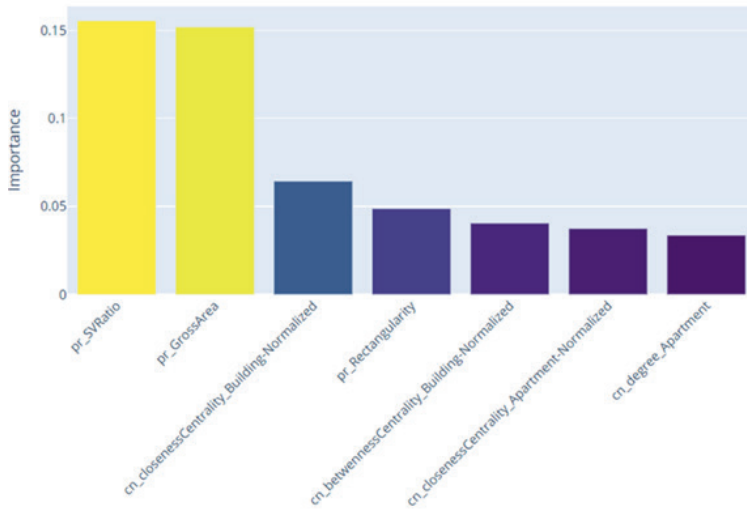


Figure 60. Results of the sensitivity analysis on node features, highlighting the relative importance of each attribute in the classification model.

Notes

¹ Supported by well-established algorithms in graph data science, graph networks excel indeed in tasks such as analysing circulation patterns, identifying central spaces, and detecting communities or zones within plans.

² Unlike private buildings of the same period, which were often documented with minimal paperwork, applications for public funding had to include extensive records demonstrating feasibility and compliance. Consequently, each analysed building was accompanied not only by plans and sections but also by metric estimates, structural drawings, technical specifications, and photographic reports. Although not directly used in this research, these documents were fundamental for attributing construction characteristics to the buildings.

³ Closed polylines were drawn to delineate the gross area of each space, and vertices were placed at the centre of each door opening to capture potential connections between spaces.

⁴ In computing, a visual programming language (VPL), is a programming language that lets users create programs by manipulating program elements graphically rather than by specifying them textually. A VPL allows programming with visual expressions, spatial arrangements of text and graphic symbols, used either as elements of syntax or secondary notation. For example, many VPLs like GH are based on the idea of “boxes and arrows”, where boxes or other screen objects are treated as entities, connected by arrows, lines or arcs which represent relations.

⁵ Boundary Representation (BRep) is a method used in computer-aided design (CAD) and solid modelling to represent the shape of a 3D object by defining its boundaries, such as surfaces, edges, and vertices. It describes an object through its outer geometry and the way these elements connect to form a solid.

⁶ For instance, an automated grid-snapping procedure was integrated into the workflow, aligning vertices to a 10 cm grid, to ensure accurate matching of the curves for computing adjacency relationships.

⁷ UCDC measures how evenly the graph splits when a node is removed; the node whose removal yields the most balanced components is assumed to correspond to the stairs.

⁸ Degree (number of direct connections) for the node in the graph.

⁹ A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

¹⁰ k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid).

¹¹ Random forests, also known as random decision forests, are an ensemble learning method for classification, regression and other tasks that work by creating a multitude of decision trees during training. For classification tasks, the output of the random forest is the class selected by the most trees.

¹² XGBoost (eXtreme Gradient Boosting) is a distributed, open-source machine learning library that utilises gradient boosted decision trees, a super-

vised learning boosting algorithm that leverages gradient descent. It is known for its speed, efficiency and ability to scale well with large datasets.

¹³ In ML, an ANN is a computational model composed of interconnected units called neurons, linked by edges that carry weighted signals. Each neuron processes input signals using an activation function and passes the output to other neurons. Neurons are organised into layers (input, hidden, and output). ANNs learn by adjusting weights during training and are widely used for tasks such as classification and regression.

¹⁴ In the field of ML and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualisation of the performance of a classification algorithm. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class, or vice versa.

6. TOPOLOGICAL GRAPHS FOR SAFETY EVALUATION OF EVACUATION NETWORKS

Abstract

Continuing the book's sequence of demonstrative case studies, this chapter presents an application in which graphs are used as the primary means to evaluate the safety conditions of building evacuation systems.

In particular, the text introduces a semi-automated workflow designed to transform a Topological Building Information Model (TBIM) into a Building Safety Model (BSM), which is structured around three complementary graphs, allowing width and length ratio safety checks to be performed at the building, storey, and room levels. The case study focuses on two university buildings, chosen for their typological complexity and relevance to occupancy-based safety analysis.

The work builds upon the research described in '*Topological BIM for Occupancy-Based Integrated Energy and Safety Analysis of Higher Education Buildings*' (Massafra *et al.* 2025a), presented at the 2025 EC3 & CIB W78 conference. It was developed within the DIGITMAN project (PRIN 2022 programme, id. 2022JM-RX2A), conducted under a collaboration between the University of Bologna, the Politecnico di Milano, and the Università Politecnica delle Marche. Special acknowledgement is extended to Gabriele Bernardini for his contribution to the development of the safety module.

6.1 Contextualization and use case definition

This chapter presents the second demonstrative application included in the book. While the preceding section focused on the use of graphs for space connectivity analysis, the attention here shifts to the use of topological networks for evaluating the safety performance of buildings' evacuation routes.

In particular, the study demonstrates how interrelated graphs can facilitate the creation of a Building Safety Model (BSM), which, when combined with graph analytics and algorithms, enables occupancy-based estimation of safety conditions by verifying the compliance of evacuation networks with the Italian Fire Prevention Code (Fuoco 2022). As in the previous case study, graphs are primarily employed in this section as tools for network analysis, rather than as means for data integration. However, this use case introduces a higher level of topological complexity, revealing additional analytical potential of graphs for building topology representation.

The research is based on a workflow that takes as input a BIM model, enriched with all necessary information for safety checks, and automatically generates three interconnected graphs representing the BSM. These are the *Passage Graph*, representing the space-door-space connectivity chains, the *Skeleton Graph*, which maps evacuation routes for width ratio verifications, and the *Isovist Graph*, which maps evacuation routes for length calculation. After being generated, these graphs are processed to assess the compliance of the building with 'deemed-to-satisfy' (DTS) requirements, performing checks that encompass various scales (i.e., fire zone, storey, room, and element scale) and analytical aspects (i.e., width ratio, length ratio, independent exits, and openings verification).

Since all these checks depend on the distribution of occupants along the evacuation routes, which essentially represent connectivity maps between spaces and doors, graph networks provide an expressive representation for this domain. Nodes denote rooms, corridors, or doors, and edges encode direct connections between them. In this way, the architectural logic of planimetric connectivity is digitally encoded in graph models, enabling the application of algorithms from graph theory, such as those used to compute shortest paths, identify bottlenecks,

and assess path redundancy.

The primary outcome involves the development of an open-BIM workflow that, starting from IFC models, integrates geometric, topological, and semantic information within a unified framework, resulting in the BSM—a graph-based model that captures different aspects of evacuation and allows for the implementation of safety checks.

6.2 Case study: safety management in public buildings

The work is applied to the field of higher education building management. The primary motivation for this choice, made during the conceptualisation of the research, was that the continuous use and high-occupancy variability typical of this functional type make these buildings particularly suitable for investigating the relationship between spatial configuration, evacuation networks, and occupancy in architecture.

Occupancy conditions, indeed, significantly influence the utilisation and management practices of such buildings, directly impacting both operational and strategic aspects, including maintenance, operation, but also safety issues. Recognising and integrating occupancy data within digital building management frameworks is, therefore, crucial for supporting strategic planning, enhancing daily building operations, and optimising the financial and technical resources allocated to real estate portfolios.

More specifically, the overall approach has been validated on a university campus affiliated with Politecnico di Milano, located in Lecco, in Northern Italy (Figure 61 and Figure 62). The campus comprises five buildings dedicated to teaching and research, with a total area of 47,000 m², 29 classrooms and 2,850 students hosted. In this text, two of them, 'Building 10' and 'Building 09', were analysed. These are standalone structures with two and three elevation levels, respectively, and an underground level, connected at the first floor through a suspended, enclosed bridge (therefore sharing the same evacuation system). The building accommodates primarily academic functions (classrooms and study rooms) and related services (restaurant and toilets).

The functional complexity of the buildings, combined with the variability of occupancy conditions and the wide availability of occupancy and geometric data on the assets, enabled the application and validation of the workflow in a real context that is also representative of a wider sample of university buildings in the country.



Figure 61. Aerial photo of the Lecco Campus (Politecnico di Milano).



Figure 62. Aerial overview of the Lecco Campus (Politecnico di Milano).

6.3 Methods and tools

6.3.1 Workflow articulation

The methodology adopted to pursue the research objectives is organised into three main phases, described in the following sections:

1. Data acquisition;
2. Data modelling;
3. Data processing and analysis.

6.3.2 Data acquisition

The data acquisition phase, undertaken by the research team at the Politecnico di Milano within the DIGITMAN project – specifically under Work Package 1 (‘Conceptual framework and database definition’) – was aimed at collecting a comprehensive set of information on the buildings under investigation. The process relied on the systematic retrieval and analysis of documentation made available by the property managers.

More precisely, the following data were gathered:

- Geometric configuration and functional layout of the two buildings, including the acquisition of CAD floor plans and spatial inventories.
- Geometric and fire performance characteristics of the evacuation systems, with particular attention to the width and height of egress components such as doors, corridors, internal staircases, external stairways, and the compartmentation features of the buildings.
- Orientation of openings along escape routes, as well as the typology of door handles and hardware associated with egress pathways.
- Geometric, physical, material, and performance attributes of the key building components, including the envelope system and internal partitions.
- Building occupancy data, as recorded by the university’s administrative office at the beginning of each academic semester.

All these data were employed in the data modelling phase, which entailed the development, first, of BIM models, then, of

dedicated BSM models, all enriched with the acquired information.

6.3.3 Data modelling

Building Information Modelling

The first step of the data modelling phase consisted of developing, for each building, the so-called Topological BIM (TBIM) model¹. This model was created using the methodology described by Massafra *et al.* (Massafra *et al.* 2024c, 2025a), briefly summarised in Figure 64.

The primary data sources used to define the spatial geometry within the TBIM models were the CAD drawings supplied by the building administrators, complemented by additional information provided by the campus management regarding the functional designation of each space. At this stage, the key input data also included occupancy parameters derived from the administration's records, specifying both the maximum and the peak occupancy levels associated with each room (see Table 11).

Once semi-automatically generated in Autodesk Revit, the TBIM models of the buildings comprised both spatial elements (spaces, storeys, zones) and interface elements (walls, floors, and roofs, windows, and doors). Minor manual adjustments were subsequently applied to specific instance objects within Revit², followed by direct export to the IFC format (Figure 63).

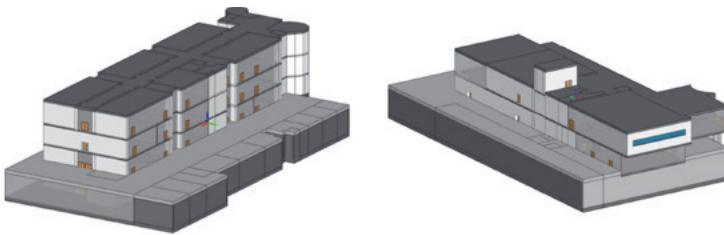
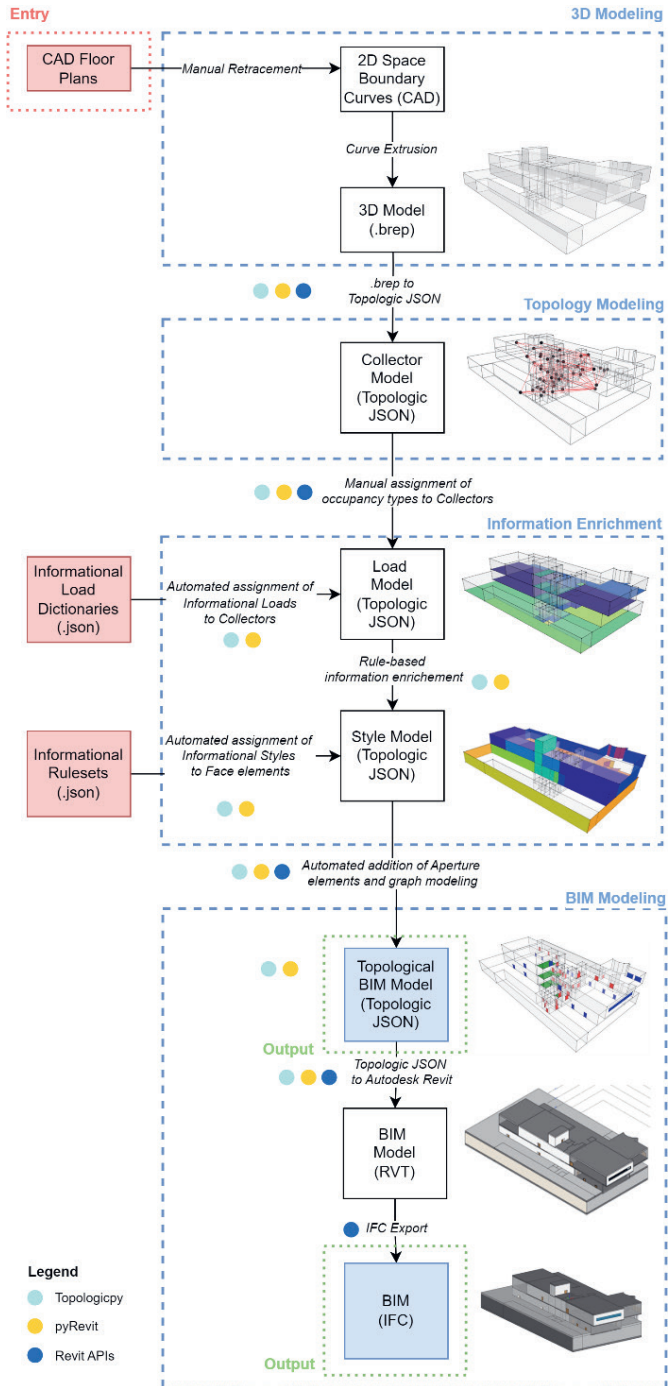


Figure 63. The TBIM models of the project's case study in the IFC format. On the left, Building 09, on the right, Building 10.

Figure 64. Summary diagram of the TBIM methodology. Please refer to the cited sources for a detailed description of the workflow (Massafra et al. 2024c, 2025a)



Property Name	Description	Quantity	Unit
Occupancy Density Safety**	Number of people required per area for the activity assigned to this space, as per fire safety regulations.	Occupancy Density	pp/sqm
Occupancy Density Peak**	The number of people estimated to be in an area of the facility during occupancy peak hours.	Occupancy Density	pp/sqm
Occupancy Number Peak**	The maximum number of people required for the activity assigned to this space in peak time.	People Count	pp
Is Occupied	Indicate whether the space is permanently occupied (TRUE) or not (FALSE).	Bool	-
Occupancy Type*	Occupancy type for this object.	Text	-

Table 11. Safety properties added to the spaces of the topological model in the information enrichment step.

** The data on occupancy types was provided by the building administration and verified through on-site inspections. Subsequently, it was aligned with the OmniClass notation.

* The data on the maximum occupancy capacity of the spaces, on the other hand, was derived, occupancy type by occupancy type, from the guidelines outlined in the Italian fire prevention code. For specific special spaces, such as classrooms, this data was adjusted to reflect the actual maximum number of occupants established by the owners.

Building Safety Modelling

After generating the TBIM in IFC format, a BIM-to-BSM process was applied to the TBIM to obtain the models required for safety verification (Figure 65). This process entails converting the IFC model into three distinct Topologic graphs that collectively represent the building's evacuation system, and that, aggregated, construct the BSM. This last model offers a comprehensive analytical framework that facilitates the execution of various safety assessments. The three types of graphs, each serving different purposes, are:

- *Passage Graph*: It maps the relationships of passage between the various spaces in the building.
- *Isovist Graph*: It helps determine the shortest path connecting occupied spaces to the nearest emergency exits on the storey, performing length checks.
- *Skeleton Graph*: It maps the flow of people through the building's evacuation routes and performs width checks.

Enriched with all necessary information, they were used to perform basic fire safety verifications in accordance with the Italian Fire Safety Code (FSC) regarding DTS solutions³.

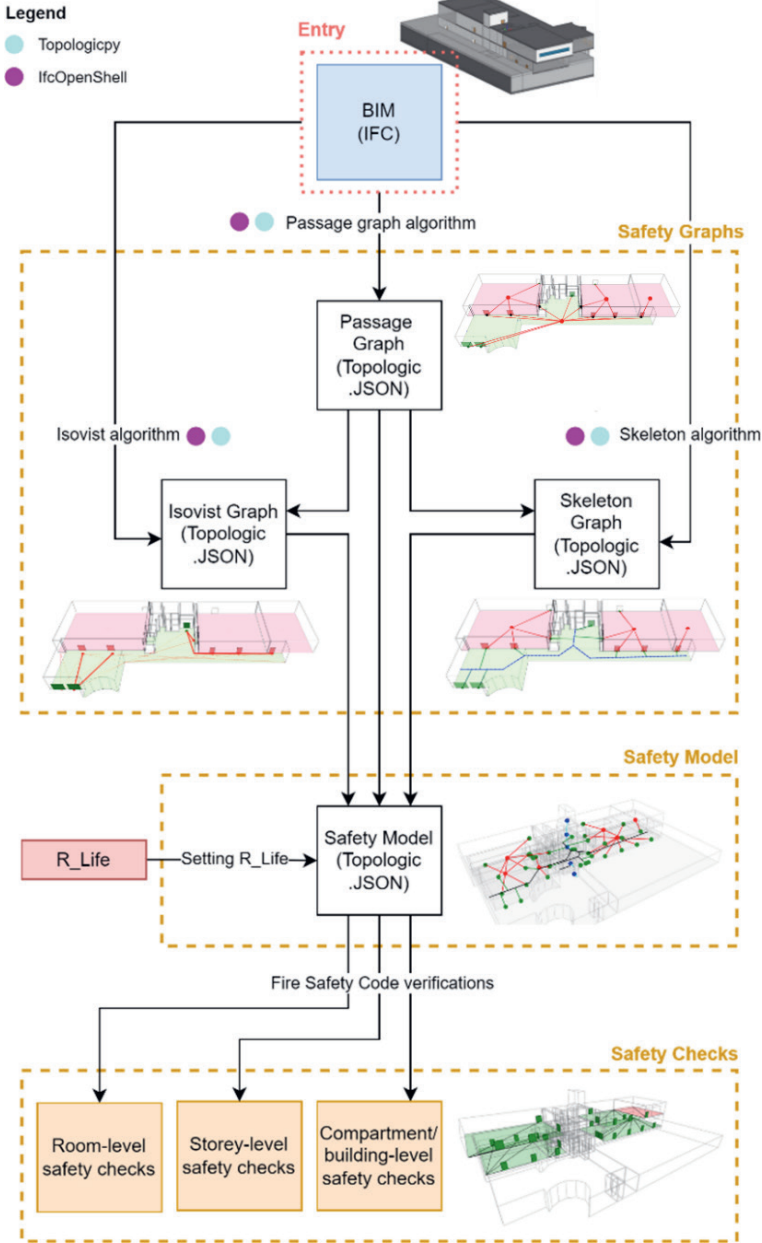
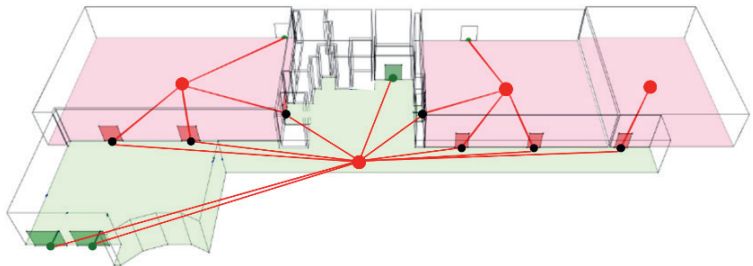


Figure 65. Summary representation of the BIM-to-BSM translation process.

6.3.4 Passage Graph

The Passage Graph maps the relationships between passages in various spaces within the building. It is an *ageometric* graph in which nodes represent spaces and doors, and edges represent direct connections between spaces and their adjacent doors⁴ (Figure 66). To construct this graph, the IFC model was processed using a Python algorithm that performed the following operations. First, the geometry of each door was imported into BRep format using `IfcOpenShell`. Afterwards, a bounding box was computed for every door using `Topologicpy`. Each `IfcSpace` object was likewise converted into a BRep representation and subsequently transformed into a `Topologic Cell`. To establish adjacency relationships, directional vectors were emitted from the bounding box of each door, oriented along the door's normal and extended in both directions. Whenever these vectors intersected a `Cell`, a connection between the corresponding door and space was identified. This relationship was then encoded in a `Topologic Graph` by creating edges linking the centroid of each door to the centroid of the intersected spaces.

Figure 66. Passage graph for the first floor of Building 10. Each space is represented as a node (in red), and each door is represented as a node as well (black for internal doors and green for external doors). The edges (in red) connect spaces to their adjacent doors, without encoding the geometry or shape of the spaces.



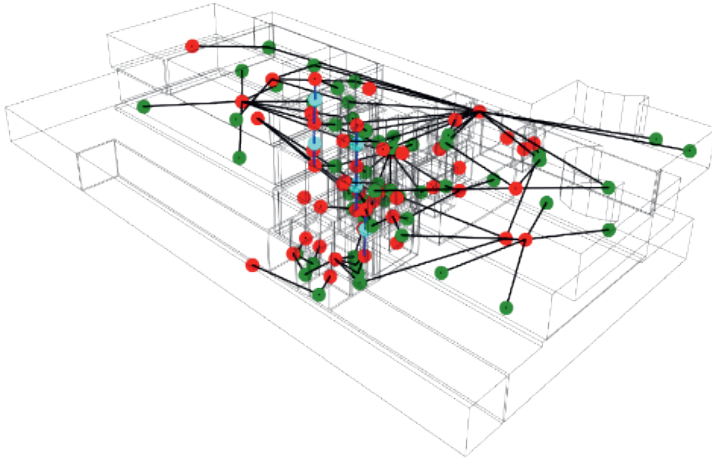
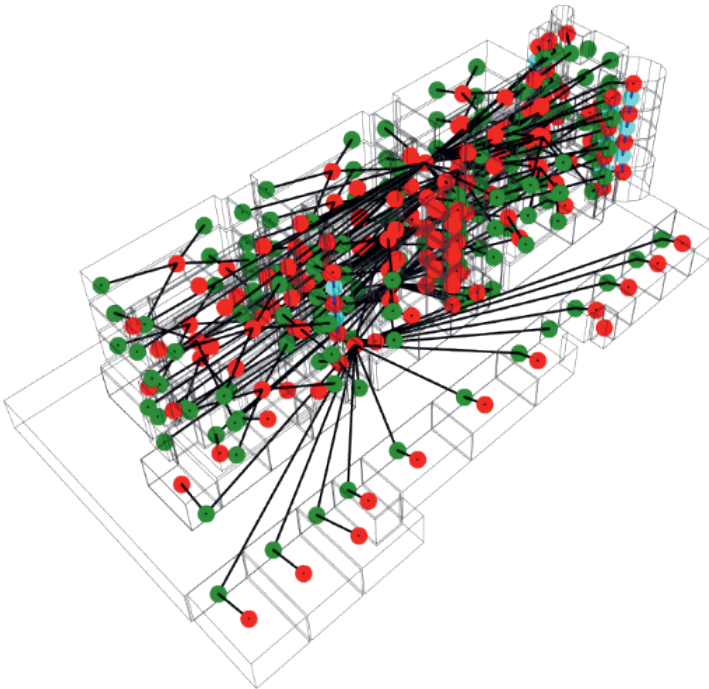


Figure 67. Passage graphs for Building 10 (on the top) and Building 09 (on the bottom).



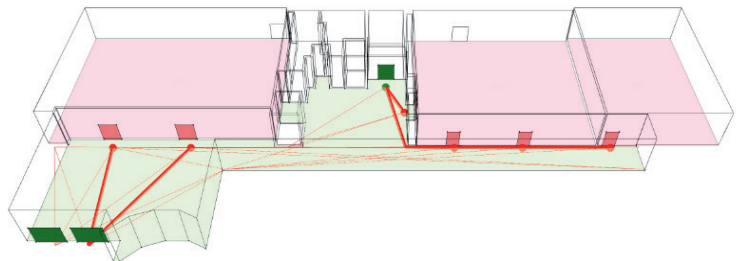
6.3.5 Isovist Graph

The Isovist Graph helps determine the shortest path connecting occupied spaces to the nearest emergency exits on the storey (Figure 68). It is called ‘isovist’ because its underlying algorithm is the isovist algorithm⁵ (Benedikt 1979), which basically corresponds to the ‘straight-line’ method required by the Italian FSC for calculating evacuation route lengths.

To generate the Isovist Graph from the IFC model, a dedicated Python algorithm was developed and applied to each building. The procedure began with the pre-processing of the IFC file, during which the model was filtered by storey and only the relevant entities, primarily *IfcSpace* and *IfcDoor* objects, were extracted. All doors on a given storey were then identified and classified into internal doors, which act as potential points of origin for evacuation, and emergency doors, which represent the target exits. For each door, its geometric representation and spatial location were retrieved for subsequent analysis.

Once the data was prepared, the isovist algorithm was applied to the geometry of each storey. This computation provided a visibility-based characterisation of the circulation system, indicating which emergency doors were visually and topologically accessible from any given internal door. The algorithm then iterated over every internal door, initiating an isovist-based search that explored the surrounding spaces and examined how visibility expanded outward from the starting point, searching for any emergency doors accessible on the storey. This search continued until at least one emergency door on the same storey became reachable, thereby establishing a viable route.

Figure 68. Isovist graph for the first floor of Building 10. Each internal door (in red) is connected to the external exits (in green) through the shortest available route (shown through red edges).



Each valid connection identified through this process, namely every traversable route linking two doors, was recorded as a set of edges. These edges were subsequently aggregated into a *Topologic Graph* that captured the complete set of isovist-based relationships between internal doors and emergency exits. The resulting graph was then processed using a shortest-path algorithm⁶ to determine the minimum-length evacuation routes for the building.

Skeleton Graph

The 'Skeleton Graph' is used to map the flow of people through the building's evacuation routes and to perform necessary checks (Figure 69), such as verifying corridor widths based on the number of people using them. Like before, this graph is named according to its underlying algorithm: the 'skeleton' algorithm⁷ (Saha *et al.* 2017).

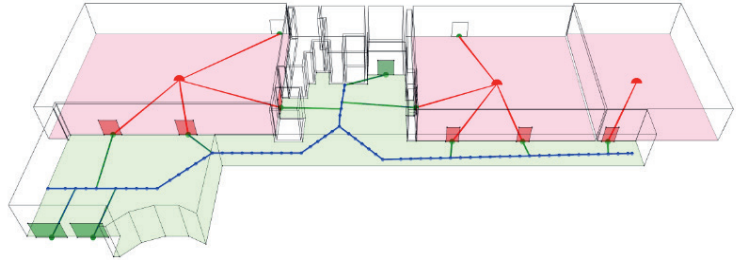
The Skeleton Graph was generated from the IFC model through another dedicated Python algorithm. This procedure was performed by processing each circulation space individually.

First, the geometry of each *IfcSpace* corresponding to a circulation area was converted into a *Topologic Cell*. From this representation, the lower face of the cell was extracted and used as the input surface for the skeleton algorithm, which produced a preliminary skeleton wire representing the medial structure of the space. The resulting skeleton wire then underwent several post-processing steps. Non-ridge wires (those not located along the main ridge line of the skeleton) were removed to retain only the primary medial axis. The endpoints of the skeleton were subsequently extended until they reached the boundaries of the corresponding *IfcSpace*, ensuring that the skeleton accurately reflected the full navigable extent of the circulation area. Finally, each internal and emergency door was connected to the nearest vertex of the skeleton, establishing explicit links between access points and the medial representation of the space.

At the end of this workflow, all generated edges were aggregated into a *Topologic Graph* for each circulation space. These space-level graphs were then merged on a storey-by-storey basis, after which inter-storey connections were incorporated through the modelling of staircases. The final output was a uni-

fied Skeleton Graph representing the complete circulation network of the building.

Figure 69. Skeleton Graph for the first floor of Building 10. Occupied spaces (classrooms) are represented as nodes in red; internal and external doors are represented as green nodes; circulation spaces (corridors) are depicted as a set of blue line segments that describe the evacuation pathway system on the floor.



6.3.6 Semantic alignment

As mentioned, during the data modelling phase, the conversion of the IFC into Topologic Graphs was achieved using a set of Python algorithms that align IFC element classes with Topologic's and with the definitions of the Italian Fire Prevention Code.

Due to the need to translate a general-purpose BIM into a discipline-specific model, shift from 3D to a graph representation, and the absence of an explicit ontology in either Topologic or the FSC, the first step in the BIM-to-BSM conversion was to create a project-specific ontology. This ontology, aligned with the FSC definitions and the Topologic class hierarchy, served as the intermediate common layer for model translation. Conceived as an extension of the Federated Ontology Framework (FOF) (Chapter 3), it also enables integration with other disciplines defined within the same FOF, as shown in later chapters.

The extension of the FOF for safety-related modelling is shown in Figure 70. This is part of a larger ontology used in the DIGITMAN project to map digital objects within graph databases.

Within this part of the ontology, spatial elements are represented, as in the FOF, using BOT's classes (`bot:Site`, `bot:Building`, `bot:Storey`, and `bot:Space`) alongside Brick classes, specifically `brick:Fire_Zone`, which aggregates multiple spaces into fire zone units. Furthermore, doors along evacuation routes are

modelled as top:Aperture instances, linked to their corresponding bot:Space entities via bot:interfaceOf relationships.

Additionally, the building’s evacuation system is mapped. The dgm:EvacuationSystem class conceptualises a network structure composed of interconnected nodes and edges – i.e., a graph – that together define one or more dgm:EvacuationRoute. Within this network, nodes represent discrete spatial entities, such as enclosed spaces (bot:Space), apertures (top:Aperture), or external areas (dgm:OpenAirSpace), while edges represent the linear connections or segments that link these nodes. Each continuous sequence of connected edges and nodes corresponding to a single evacuation path – such as the route connecting one room to another – is defined as a dgm:Route. The combination of multiple dgm:Route instances collectively forms the complete evacuation network, representing the overall dgm:EvacuationSystem of the building.

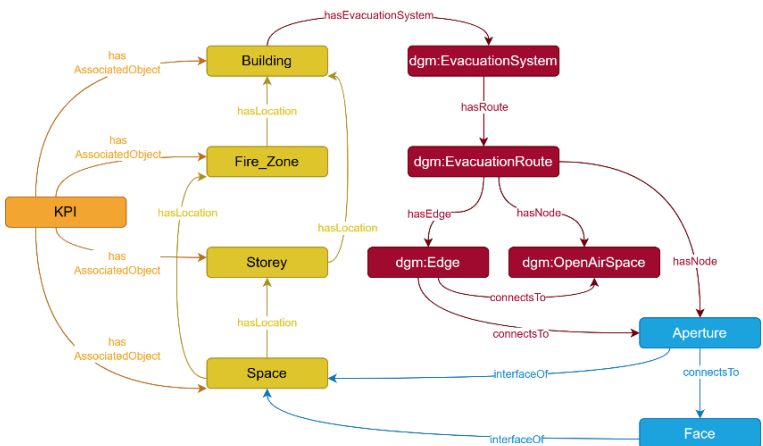
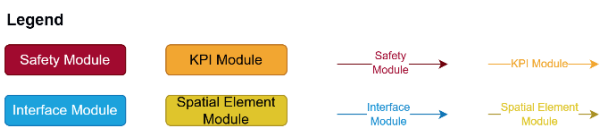


Figure 70. Extension of the Federated Ontology Framework for modelling safety-related data.



6.3.7 Data processing and analysis

Once modelled, the three graphs were used to perform the checks listed below on DTS solutions, in accordance with the Italian FSC. These verifications are commonly performed at multiple levels, ranging from the entire building's evacuation system to individual doors within it, and depend on a single parameter: the risk profile for human life safety (R_{life}). This parameter, to be assigned to each fire zone, implies different levels of assignment criteria for the performance of evacuation routes and thus specific DTS solutions⁸. In the context of this study, safety calculations were run at four different levels:

- *Element-level verifications* refer to checks of the size, particularly the width, of the doors along the evacuation routes (doors).
- *Space-level verifications* refer to checking the minimum number of independent exits from a space and their width, intended to prevent localised overcrowding effects of doors as bottlenecks⁹. Additional space-level verifications refer to the door handling typology, opening direction (UNI EN 1125/UNI EN 179), and the minimum height of escape routes.
- *Storey-level verifications* are based on the element-level verifications, which are aggregated at the storey level. They require checking the elements of horizontal escape routes (i.e., rooms, corridors, and doors along them, up to the staircases), including their minimum width (including redundancy verification) and height¹⁰.
- *Compartment- and building-level verifications* mainly concern vertical routes. For staircases, verification of the minimum width is performed using the same rationale as for horizontal escape routes and doors, including a redundancy check¹¹. Moreover, compartment-level verifications encompass overall verifications of the entire maximum route length, including all horizontal, vertical, and dead-end sections¹².

In the following paragraphs, the results of applying the previously presented methodology to the project's case study are reported. The application aimed to calculate safety KPIs to perform safety assessments at the space, storey, and compartment/building levels.

6.4 Demonstration

6.4.1 Application to the room

The demonstration of the proposed methodology on the project's case study began with the application of a first verification algorithm that, using Python, processed the BIM and BSM and automatically checked the compliance of individual occupied spaces in the models (e.g., classrooms and offices).

More specifically, the algorithm, applied to both buildings, allowed checking whether each space met the FSC's criteria, including the width ratio¹³ of the travel path, the number of independent exits of the space¹⁴, and door opening specifications¹⁵. The workflow was applied in redundancy conditions¹⁶.

The process calculated relevant KPIs at the room level and generated visual reports to help identify potential safety issues (see the example in Figure 71).

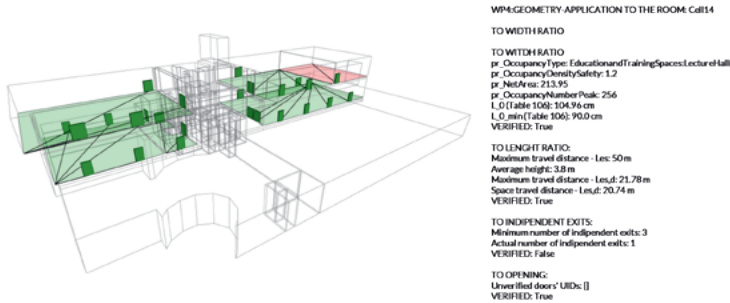


Figure 71. Room-level verifications conducted on Building 10 under a fictional occupancy scenario. On the left, 3D view. On the right, report of the most significant parameters considered in the evaluation for a space.

6.4.2 Width ratio verifications

After checking single rooms, the verifications were extended to the storey and compartment level, starting from the width assessment and continuing with the length assessments.

The width assessments were conducted based on the Skeleton Graph. Three preliminary steps were required for preparing the BSM for running the analysis:

1. Conversion of the Topologic graphs into a NetworkX graph, necessary to speed up shortest path calculations¹⁷;
2. Identification of the shortest paths in the Skeleton Graph, identified as the shortest routes leading from each door ad-

- adjacent to an occupied space (e.g., classrooms) to the closest storey exit;
3. Distribution of the number of evacuees from occupied spaces along the shortest routes in the Skeleton Graph.

Identifying the shortest routes along the Skeleton Graph

To identify the shortest routes on the Skeleton Graph, all doors adjacent to both an occupied space and a corridor (i.e., the ‘start doors’) were identified by executing topological queries on the Skeleton Graph. Then, for each start door, the closest storey fire exit (‘end door’) was determined by applying the shortest path algorithm on the Isovist Graph. Finally, for each start door, the shortest path to the corresponding end door (the ‘evacuation route’) was identified within the Skeleton Graph by applying the shortest path algorithm.

The results for the case study are displayed in the following figures (Figure 72, Figure 73).

Distributing the number of people along the evacuation network

Then, evacuees were distributed along the shortest path within the evacuation network. For each start door, the number of evacuees passing through was determined by dividing the peak occupancy of the adjacent occupied space evenly among the space’s start doors. Then, the evacuation route from each start door to its corresponding end door in the Skeleton Graph was identified. At the ground level, for each door adjacent to the stairs, the shortest path from this door to the closest storey exit door was also determined. For each evacuation route, at each node along the route, the number of evacuees exiting from the start door was assigned to the corresponding node in the evacuation network. The edges of the Skeleton Graph were subdivided to allow assessments at every meter and at each intersection between different edges. Finally, on the ground floor, evacuees descending from upper floors via staircases were distributed along the shortest routes intersecting the evacuation paths leading from the staircases to the nearest storey exit. The results of this process for the case study are displayed in the following figures (Figure 74, Figure 75).

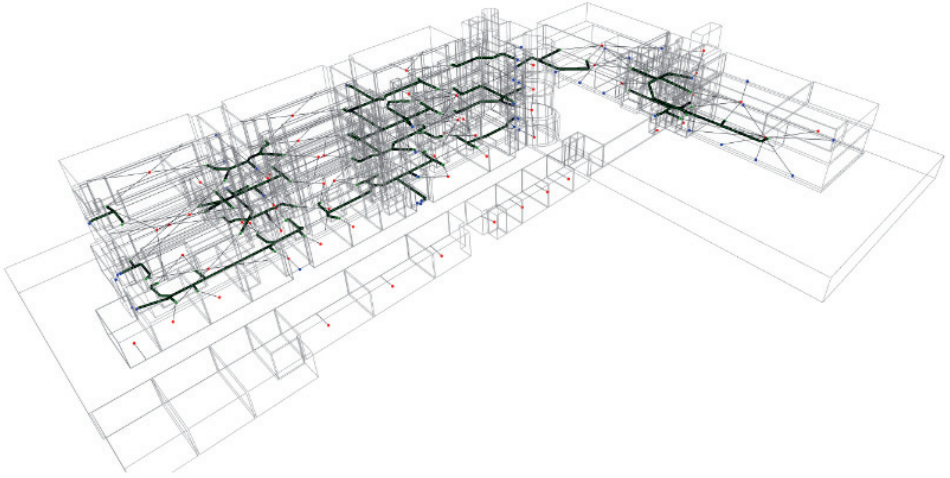


Figure 72. Combined Skeleton Graph of Building 09 and Building 10 with the shortest routes highlighted.

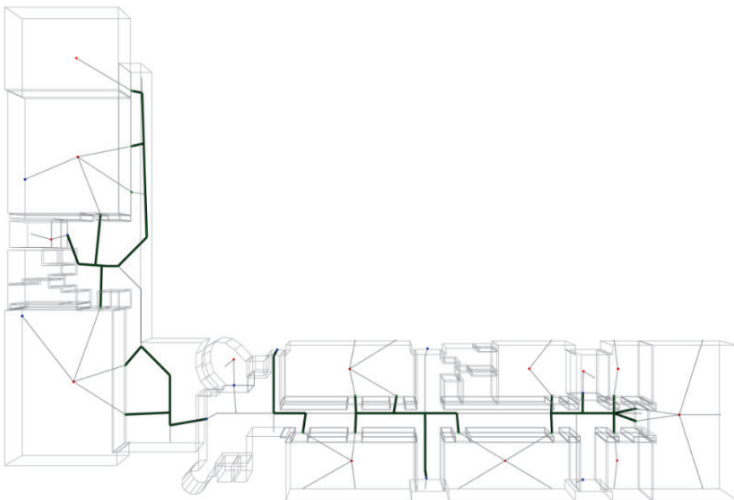


Figure 73. Combined Skeleton Graph of Building 09 and Building 10 with the shortest routes highlighted, first floor.

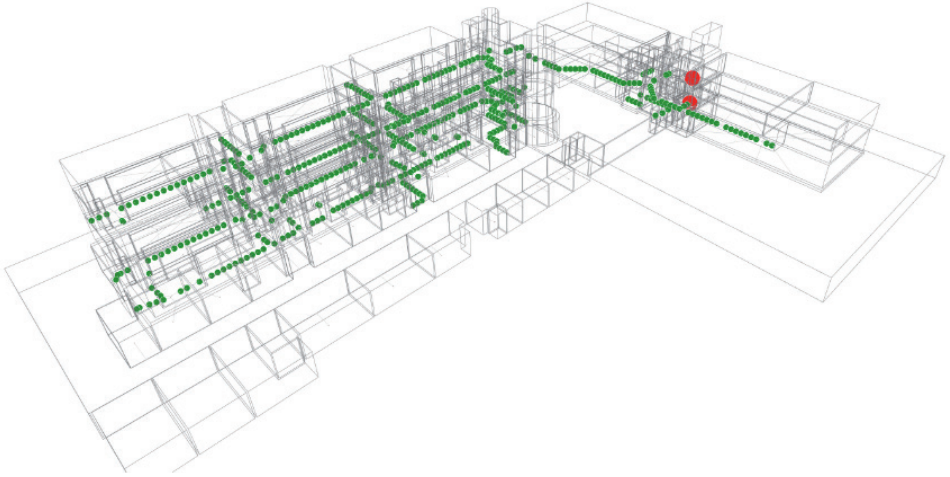


Figure 74. Combined Skeleton Graph of Building 09 and Building 10 with number of people distributed along the routes.

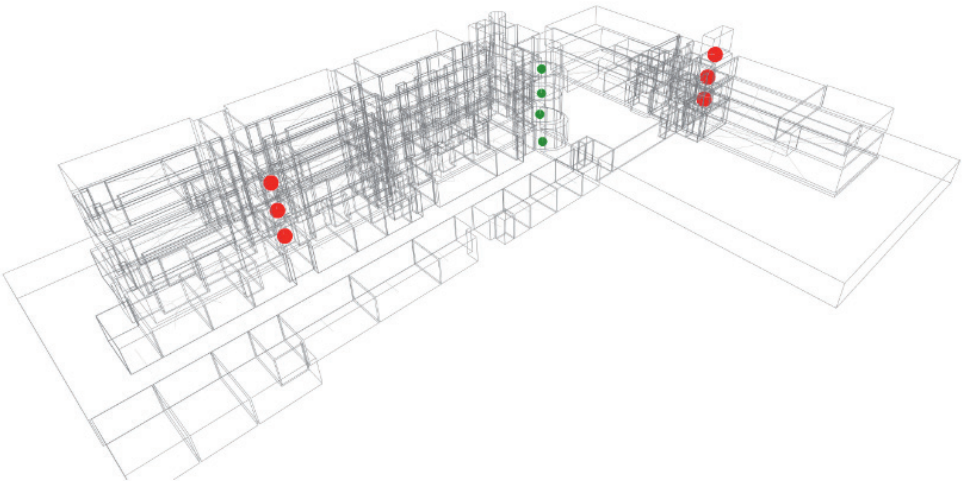


Figure 75. Combined Skeleton Graph of Building 09 and Building 10 with the number of people distributed along the routes, first floor.

Element-level verifications on width ratio

After distributing the number of occupants across the nodes of the Skeleton Graph, width verification was performed, beginning with the assessment of horizontal escape routes at the level of individual elements such as doors and corridors.

The effective passage width (L_{elem}) for each corridor vertex in the evacuation network was calculated using functions from the TopologicPy library¹⁸. The effective width of each door was instead taken from the 'width' property defined in the BIM model and assigned to the corresponding node in the Skeleton Graph.

Once the effective widths were established, they were compared with the minimum requirements defined in Table 107 of the FSC. Where compliance was confirmed, a further verification was performed based on occupant load, using the provisions of Table 106, which defines the minimum width as a function of the number of occupants by multiplying the unit width per person. An element was considered compliant only if both checks were satisfied. Finally, each element was assigned a KPI expressing the width ratio, defined as the ratio of the minimum width required by the FSC to the door's effective width (Figure 76).

A similar algorithm was applied to verify the minimum stair widths. In this process, vertical circulation spaces were first identified as subgraphs composed of interconnected rooms classified under the 'stairway' function. The number of occupants assigned to each stairwell was then determined in accordance with the adopted evacuation strategy, which could be either 'simultaneous' or 'phased', in line with the FSC's prescriptions.

Once defined, the effective width of each staircase was checked against the minimum requirement specified in Table 111 of the FSC. Where this condition was satisfied, a further verification was conducted based on occupant load, comparing the effective width with the required value in Table 108, which expresses the minimum width as a function of the number of evacuees. For each stairway, the width ratio was then calculated and assigned to the corresponding node within the Skeleton Graph. The results are shown in Figure 77.

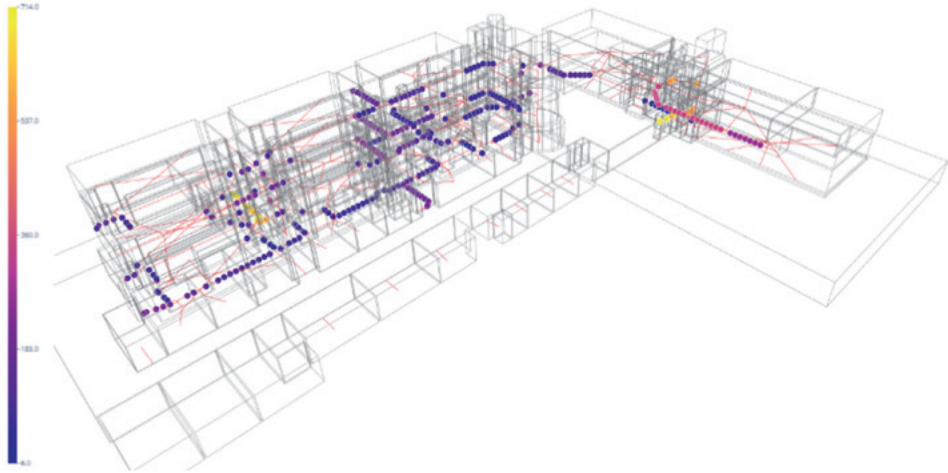


Figure 76. Verification of the minimum width of horizontal escape routes on the combined Skeleton Graph under a fictional occupancy scenario.

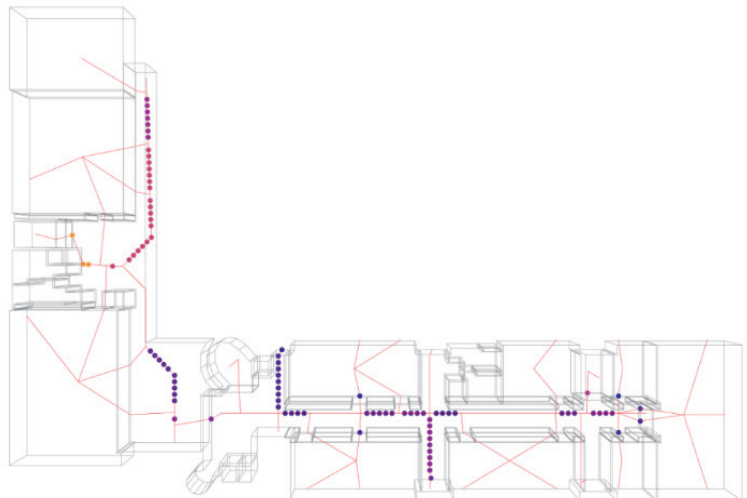


Figure 77. Verification of the minimum width of vertical escape routes on the combined Skeleton Graph under a fictional occupancy scenario.

Storey-level verifications on the width ratio

The width ratios of horizontal escape routes were then aggregated at the storey level.

This process began with the identification of all storey exits within each compartment, including those leading to safe or exterior areas, as well as those connecting corridors to internal staircases. The total effective width of these exits was calculated by summing the actual widths of all identified openings. In parallel, the total minimum width requirement for each storey was determined using the same calculation principles applied in the previous verifications. Finally, the ratio between the cumulative minimum required width and the cumulative actual width was computed, providing the storey-level KPI for horizontal escape route capacity. This indicator provided a concise measure of compliance for each storey, indicating whether the available egress capacity met or exceeded the FSC standards.

Figure 78 visually represents the results of this process applied to the project's case study.

Compartment-level verifications on the width ratio

The final width verification was performed at the compartment level for the building's final exits located on the ground floor. This verification considered not only horizontal evacuation flows but also vertical flows coming from the staircases.

The verification process strictly follows Formula 17 of the FSC. This ensures that the cumulative evacuation capacity of all final exits is sufficient to accommodate the total number of evacuees from both horizontal and vertical escape routes.

Figure 79 depicts an output of this process applied to the project's case study.

Redundancy

A redundancy check was finally applied to all width verifications to ensure minimum width compatibility under failure conditions. This check evaluates the evacuation capacity by considering one exit unavailable at a time. The objective was to verify whether the remaining exits can still accommodate the required evacuation flow, ensuring compliance with safety regulations even in the case of an exit blockage or failure.

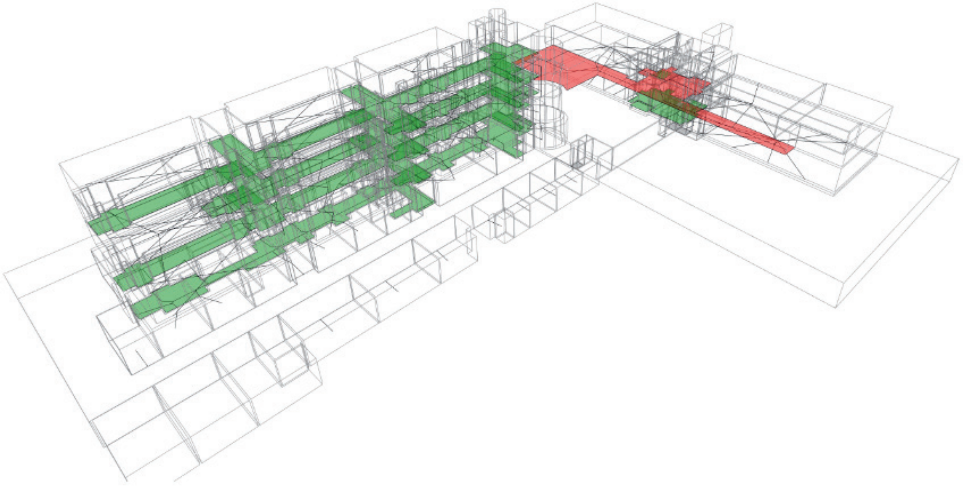


Figure 78. Verification of the minimum width of horizontal escape routes at the storey level for Building 09 and Building 10 under a fictional occupancy scenario.

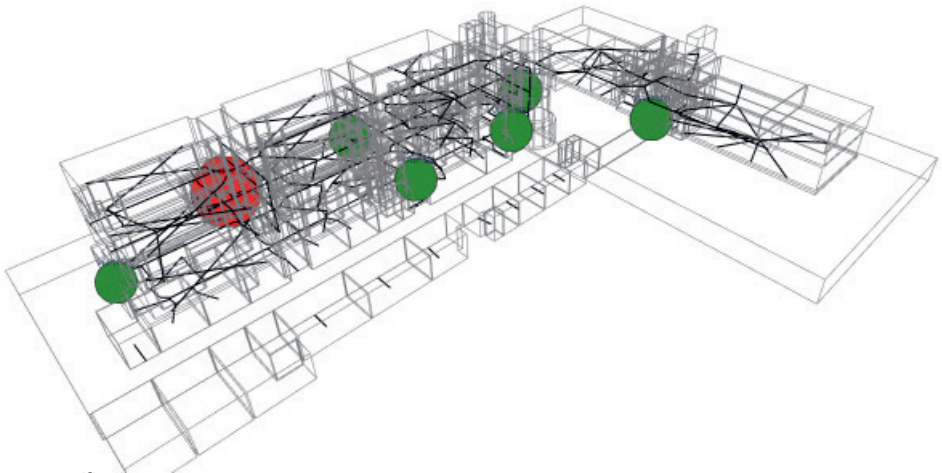


Figure 79. Verification of the minimum width of final exit doors on the compartment level for Building 09 and Building 10 under a fictional occupancy scenario.

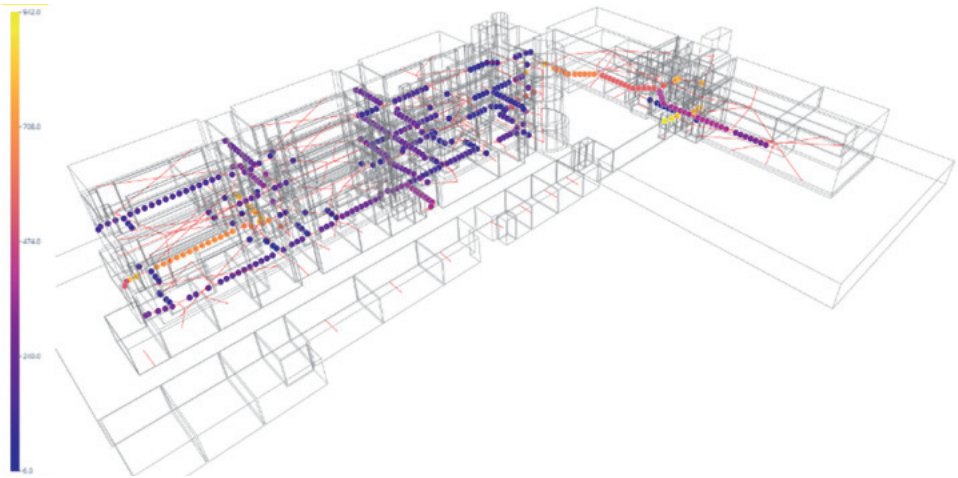


Figure 80. Combined Skeleton Graph of Building 09 and Building 10 with the number of people distributed along the routes after applying redundancy.

6.4.3 Length ratio verifications

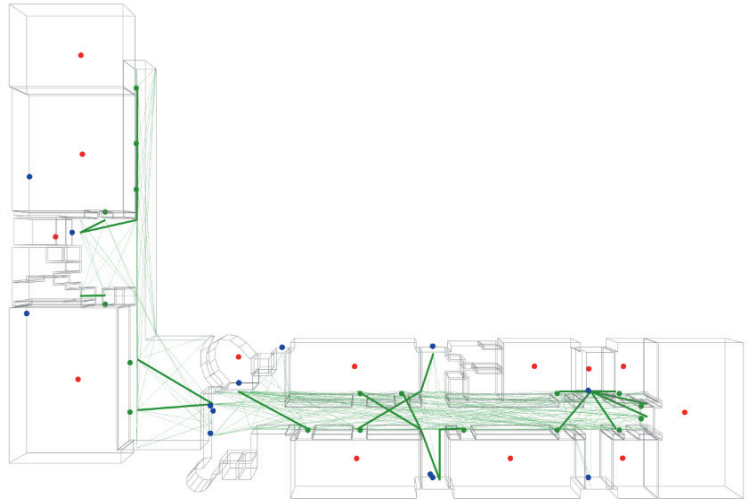
Identifying the shortest routes along the Isovist Graph

After width checks, length checks were performed. To identify the shortest evacuation routes on the Isovist Graph, all doors adjacent to both an occupied space and a corridor were first identified via topological queries and designated as start doors. For each of these, the nearest storey fire exit was determined by applying the shortest-path algorithm directly to the Isovist Graph. Once the start and end doors were established, the corresponding evacuation routes were traced within the Skeleton Graph, again using the shortest path algorithm to identify the optimal paths connecting each pair of doors. This process ensured that the evacuation network identified the most efficient routes from occupied areas to designated exits (Figure 81).

Calculating the actual length of the evacuation routes

The actual lengths of the evacuation routes were calculated by combining the paths from occupied spaces to the start doors, the horizontal connections, and the vertical circulation routes¹⁹. The total effective evacuation path length was obtained by summing the distances of all horizontal and vertical segments, along with those from occupied spaces to their start doors, enabling the automatic identification and evaluation of both dead-end and continuous corridors.

Figure 81. Combined Isovist Graph of Building 09 and Building 10, shortest routes (first floor).



Calculating the maximum length of the evacuation routes

The maximum length of the evacuation routes was calculated, for each case, following the guidelines in Table 104 of the FSC (maximum travel distances) and the R Life category. Adjustments were considered to increase the maximum travel distance in Table 104 by applying the factors in Table 117, which account for additional fire protection requirements for the compartment served by the escape route.

The final length-to-width ratio for each space was calculated by selecting the shortest available evacuation path and dividing its actual length by the maximum allowed length specified by the code.

6.5 Results and outcomes

The proposed workflow was implemented across the evacuation systems of Building 09 and Building 10. To verify the robustness and adaptability of the methodology, four simulation scenarios were carried out, each representing different occupancy and functional conditions.

- *Scenario 0* corresponded to the current operational state (2024) of the buildings and served as the baseline for comparison.
- *Scenario 1* modelled the case of classroom use with a 20% increase in occupancy, assessing how higher user density influences safety conditions.
- *Scenario 2* explored the opposite condition, simulating classroom use with a 20% reduction in occupancy, to evaluate the system's sensitivity to lower occupancy.
- *Scenario 3* represented an office configuration (i.e. classrooms were all refunctionalised as offices), allowing an assessment of performance under a distinct functional use with different occupancy patterns.

The assessment revealed that all configurations met the regulatory requirements, except for Scenario 1, which simulated a 20% increase in classroom occupancy. The baseline condition (Scenario 0) for the layout of lecture halls with fixed seating was fully compliant, as expected, since the existing configuration complies with the FSC. Similarly, the lower-density scenarios (2 and 3) were also compliant, as the reduced occupant load maintained adequate egress widths and compartmentation strategies.

By contrast, Scenario 1 triggered a safety alert. The increased occupancy overloaded several components of the evacuation network, whose exit widths were insufficient to accommodate the higher density. While some rooms could technically accommodate the additional occupants, the scenario cannot be generalised to the entire building without architectural modifications. Consequently, the model indicated several strategic options:

1. discarding the scenario entirely in cases of general occupancy increase;
2. limiting the capacity increase to non-critical spaces identi-

- fied by key performance indicators (particularly in Building 10);
3. introducing physical interventions such as widening or adding doors;
 4. adopting management-based measures for evacuation control, including directing occupants along less congested routes.

Fire safety experts should further examine these latter options to ensure compliance with regulatory requirements.

The workflow demonstrates that the proposed model can rapidly provide a KPI-based overview of safety performance, supporting informed decision-making and enabling iterative updates to the BIM and topological models at the component level according to revised occupancy and dimensional parameters.

Notes

¹ TBIM constitutes an alternative modelling paradigm to conventional, product-oriented BIM approaches. It conceptualises buildings through the aggregation of spatial elements rather than through detailed representations of construction components. This shift towards a space-based and topologically driven representation yields models that are geometrically lightweight yet semantically rich, thereby facilitating their integration into Building Performance Simulation (BPS) workflows. The TBIM methodology relies on topological and conditional modelling principles that support the semi-automated generation of structured building models, enabling a more efficient translation of architectural and functional information into simulation-ready datasets. In this regard, TBIM enhances interoperability between BIM environments and simulation tools by reducing geometric complexity while preserving the essential spatial and functional characteristics needed for performance assessments. TBIM is operationalised in Autodesk Revit using an ad-hoc developed pyRevit toolkit (Massafra et al. 2024).

² For example, TBIM apertures, initially positioned at the centre of wall faces by default in the TBIM toolkit, were manually adjusted to reflect their actual locations, and any inconsistencies in the assignment of construction data to faces or apertures were corrected. Moreover, properties that could not be represented on a functional basis through the so-called Informational Load Dictionaries, and therefore required explicit specification at the level of individual spaces, were manually added to the model.

³ Deemed-to-satisfy (DTS) refers to prescriptive compliance pathways in which fire safety requirements are fulfilled by adhering to predefined technical specifications, geometrical constraints, and design rules established by a code. When these prescriptive conditions are met, the solution is automatically considered compliant, without the need for further performance-based analysis or simulation.

⁴ In this context, 'ageometric graph' means that the graph does not encode any explicit geometric information such as dimensions, distances, or spatial coordinates. Instead, it represents only topological relationships: nodes correspond to spaces or doors, and edges denote direct adjacency between a space and its connected doors.

⁵ An isovist is the set of all points in space visible from a given observation point, considering obstacles that obstruct the line of sight. The isovist algorithm is the computational procedure used to generate this visibility polygon (or visibility volume in 3D) by casting rays outward from an observer location until they intersect surrounding boundaries. The resulting geometric region represents the observer's instantaneous field of view and encodes both spatial extent and directional characteristics of visibility. Isovist computations are widely used in spatial analysis, wayfinding studies, evacuation modelling, and space syntax research because they quantify how geometry influences perception, orientation, and movement.

⁶ A shortest-path algorithm is a computational method used to determine the minimum-length route between two nodes, where length may be defined in terms of distance or any other weighted metric assigned to the edges. It evaluates the connectivity structure of the network and identifies the path that minimises the cumulative weight from a specified origin to a target node.

⁷ A skeleton algorithm (often referred to in the context of skeletonisation) is a computational method that extracts a reduced representation, known as the 'skeleton', of a geometric or spatial object, such that the skeleton preserves the topological and structural characteristics of the original form while significantly reducing its complexity. Typically, the skeleton is obtained by thinning or collapsing the object's geometry inwards (e.g., via medial axis transformation or other contraction techniques), resulting in a set of lines or curves (in 2D) or surfaces/curves (in 3D) that reflect the object's connectivity or structural core. The skeleton algorithm thus provides a concise graph-based or curve-based abstraction of the original shape, making it suitable for analysis tasks such as routing, object recognition, evacuation modelling, and spatial navigation. In particular, the algorithm can be used to transform complex spatial geometries into graph structures by identifying nodes (branch points) and edges (connectivity paths) that correspond to the underlying geometry's topology (Koehoorn et al. 2017).

⁸ In particular, Rlife depends on the combination between the compartment/building intended use and the hazard conditions. It can be assigned according to a tabular approach provided by the Code itself. Since it provides significant examples of Rlife that can be directly used by designers, it is as easy to apply as it is for low-expert technicians. Additional verifications regarding the variation in occupancy rate and the presence and rate of vulnerable users (e.g., those with motion/sensory disabilities), as well as the fire load rate, could also be used to vary Rlife and thus evaluate the effectiveness of solutions under alternative building use conditions without layout variation.

⁹ This calculation accounts for system redundancy by assuming that, at any given time, one door is intentionally treated as unavailable.

¹⁰ Since occupants can enter the escape routes from different initial rooms, the occupancy is evaluated as occupants' sum by route section. Dead-end route sections are considered if present.

¹¹ The unit width per person is here considered based on the evacuation method ('simultaneous evacuation' or 'phased evacuation'), which varies the simultaneous maximum occupants number in the staircases.

¹² Since a building could be composed of one or more compartments, such verifications are also referred to at the compartment level.

¹³ The minimum width of each exit, to prevent localized overcrowding effects of doors as bottleneck, is derived by the occupancy assessment in number of persons in the given space according to the FSC (see Tables 106, 107, 108, 111 of the FSC (Fuoco 2022)).

¹⁴ Two door/exits are independent if, in the room, points exist where the view angle between straight routes $\geq 45^\circ$ and/or they are separated by fire-resistant elements. The minimum number of independent door/exits depends on Rlife and on the crowd size (see Table 94 of the FSC (Fuoco 2022)).

¹⁵ Verifications on the door handling typology and opening direction ask for UNI EN 1125/UNI EN 179 opening devices and opening direction in the direction of evacuation depending on occupancy > 5 persons and any specific risk in the room.

¹⁶ Redundancy check aims to verify minimum-width compatibility by considering an exit unavailable at a given time.

¹⁷ NetworkX proved faster than Topologic when applying graph algorithms (e.g., shortest-path calculation) on large graphs.

¹⁸ For each vertex, the circulation space in which it was located was identified through the `Vertex.IsInternal()` method, after which a line orthogonal to the edge of the skeleton graph was constructed and intersected with the `Brep` of the corresponding `IfcSpace` using the `Topology.Intersect()` function. The distance between the two intersection points was taken as the corridor width at that vertex.

¹⁹ The distances from occupied rooms to their respective start doors were computed using the same algorithms applied at the room level. The horizontal routes were measured as the sum of the edge lengths within the isovist graph traversed from each start door to its corresponding end door. For the vertical routes, the total length was derived by considering the height of the staircase shafts and applying trigonometric relationships to estimate the hypothetical flight length, assuming a 30 cm tread and a 15 cm riser (a 50 % slope). Landing lengths were also incorporated, with landings assumed to be as wide as the doors adjacent to the stair shafts and with two flights of stairs per floor by default.

7. SEMANTIC GRAPHS FOR DATA INTEGRATION IN HERITAGE BUILDING CHARACTERISATION PROCESSES

Abstract

The characterisation of the built heritage represents the first step for analysing its performance and implementing data-driven renovation strategies. However, this process requires the acquisition of vast, heterogeneous datasets, often owned by multiple stakeholders, which can lead to inconsistent knowledge bases.

The research presented in this chapter introduces the use of *semantic graphs* to integrate heritage building data relevant for energy performance analysis. A Knowledge Graph (KG) architecture, coupled with a time-series database, is implemented to link static and dynamic data for a couple of buildings located within the Parco Archeologico dell'Appia Antica in Rome (Italy), resulting in a web application that enables navigation of building information, visualisation of data trends, and querying relevant KPIs.

This research was conceived within the BETWIN project (AlmaValue 2023 call) and developed through a scientific collaboration agreement between the Department of Architecture of the University of Bologna, the Parco Archeologico dell'Appia Antica (PAAA) in Rome, and the Italian National Agency for New Technologies, Energy and Sustainable Economic Development (ENEA). Special acknowledgement is extended to Ugo Maria Coraglia, Silvia Di Turi, Domenico Palladino, and Michele Reginaldi.

7.1 Contextualisation and use case definition

Following two chapters that demonstrated the use of graphs for the analysis of network structures, this chapter introduces the book's first case study that focuses on the application of graphs as instruments for data integration. The text below illustrates how semantic graphs can be deployed in contexts where heterogeneous data need to be consolidated into structured knowledge, which is valuable for informing decision-making processes. Specifically, the chapter focuses on integrating the extensive and diverse datasets that typically characterise the energy assessment of heritage buildings.

Gathering reliable and structured information to evaluate the energy performance of historic buildings is essential for developing management strategies that enhance efficiency while preserving heritage significance. However, obtaining precise data in such settings is often challenging, especially when thoroughly characterising construction elements and mechanical systems.

Unlike modern constructions, historic buildings typically feature unique and complex components, obsolete building techniques, and traditional materials whose thermal behaviour is not easily predictable using contemporary standard models¹. As a result, significant challenges emerge during data collection: much of the required information is difficult to access, laborious to gather, or, when it does exist, scattered across multiple stakeholders in various formats, resulting in fragmented, disconnected data silos (Merino *et al.* 2022). Consequently, whenever building managers require this information – for instance, to perform energy audits or plan renovation works – they must obtain it from numerous stakeholders. This process is typically handled through multiple email exchanges or verbal requests, often leading to lost information and frustration, as well as considerable time and resource waste during communication (Abuimara *et al.* 2021b). Additional problems arise because performance data is not static and varies over time. This dynamism may not be compatible with the usual slowness of the data collection process.

To address these issues, this chapter introduces a knowledge-modelling workflow that links building information gath-

ered from multiple sources and in diverse formats, thereby creating robust knowledge bases that facilitate efficient data querying and retrieval to support building management activities. Central to this methodology is the Federated Data Modelling (FDM) approach presented in Chapter 3, which was operationalised through the BTwin toolkit (Chapter 4).

The research method was applied in a real environment, namely the PAAA's portfolio in Rome (Italy). In previous work, Di Turi *et al.*, from ENEA, compiled a comprehensive dataset for part of this portfolio, combining geometric surveys, non-destructive tests, in-situ investigations, environmental monitoring, and energy bills (Di Turi *et al.* 2023). This dataset enabled the evaluation of the envelope's thermal behaviour, indoor environmental conditions, and operational performance, forming a solid foundation for conducting energy audits. In this study, the data are organised within a KG to power a user-friendly web application that displays KPIs through intuitive dashboards. The result is a prototype platform that enables managers to easily access and extract the information needed for performance-based decision-making.

7.2 Case study: heritage building characterisation

The case study focuses on two sites within the Parco Archeologico dell'Appia Antica, a peri-urban protected area in Rome. The first, the Capo di Bove complex (Figure 82), comprises two buildings: a main building, primarily used as the PAAA's headquarters and as a conference hall, and a smaller outbuilding serving as the park's ticket office. The second site, Villa dei Quintili (Figure 83), includes a farmhouse currently used as office space and an adjacent antiquarium with museum functions.

These complexes were retained as of particular interest for the study because of their relevance as medium-sized protected historic complexes and the opportunity to collaborate with building administrators during data collection. The significance of this case study is further reinforced by the forthcoming restoration and retrofit interventions, funded through Italy's National Recovery and Resilience Plan (NRRP). These initiatives aim to enhance both the structural integrity and energy effi-

ciency of the buildings, requiring a comprehensive understanding of their characteristics and a systematic approach to data collection to monitor their transformation and evolving performance over time.

In recent years, ENEA has collected a wide range of data to support the performance characterisation of both envelope and system components (Di Turi *et al.* 2023, 2024). Archival documents provided partial evidence of the original layouts; on-site surveys recorded subsequent modifications and current conditions; energy bills revealed consumption patterns; thermo-fluximetric analyses documented the envelope's thermal properties; spreadsheets described the configuration of technical systems; and monitoring campaigns supplied indoor environmental data, including temperature and humidity.



Figure 82. The Capo di Bove complex: main building. Source: Di Turi *et al.* (Di Turi *et al.* 2023)



Figure 83. Villa dei Quintili's complex: main building. Source: Di Turi *et al.* (Di Turi *et al.* 2023)

However, despite the high significance and accuracy of this data, the heterogeneity of these sources made data collection particularly complex, thereby justifying the adoption of a semantic graph approach to link data into a standard knowledge base usable for planning the renovation interventions on the buildings.

7.3 Methods and tools

7.3.1 Workflow articulation

The methodology adopted to pursue the research objectives, depicted in Figure 84, is organised into five main phases, described in the following sections:

1. *Data acquisition*: gathering and collecting all primary datasets;
2. *Data modelling*: creating the HBIM model and preprocessing the time-series data;
3. *Data processing*: aggregating raw data and computing the relevant KPIs;
4. *Data integration*: linking the datasets through the federated KG;
5. *Data visualisation*: presenting performance information through interactive dashboards.

7.3.2 Data acquisition

The data acquisition phase, conducted by the ENEA research team, brought together a wide range of information needed to characterise the PAAA's buildings for an energy audit, which was performed in accordance with ISO 52016 (Di Turi *et al.* 2024). Specifically, the following data were gathered for the two sites analysed:

- *Geometric and functional data*: 2D drawings from the administrators were digitised in CAD to document the buildings' geometry, carefully verifying discrepancies between actual conditions and available documentation. Information about building use was also collected through in-situ surveys and interviews with the site administration, and

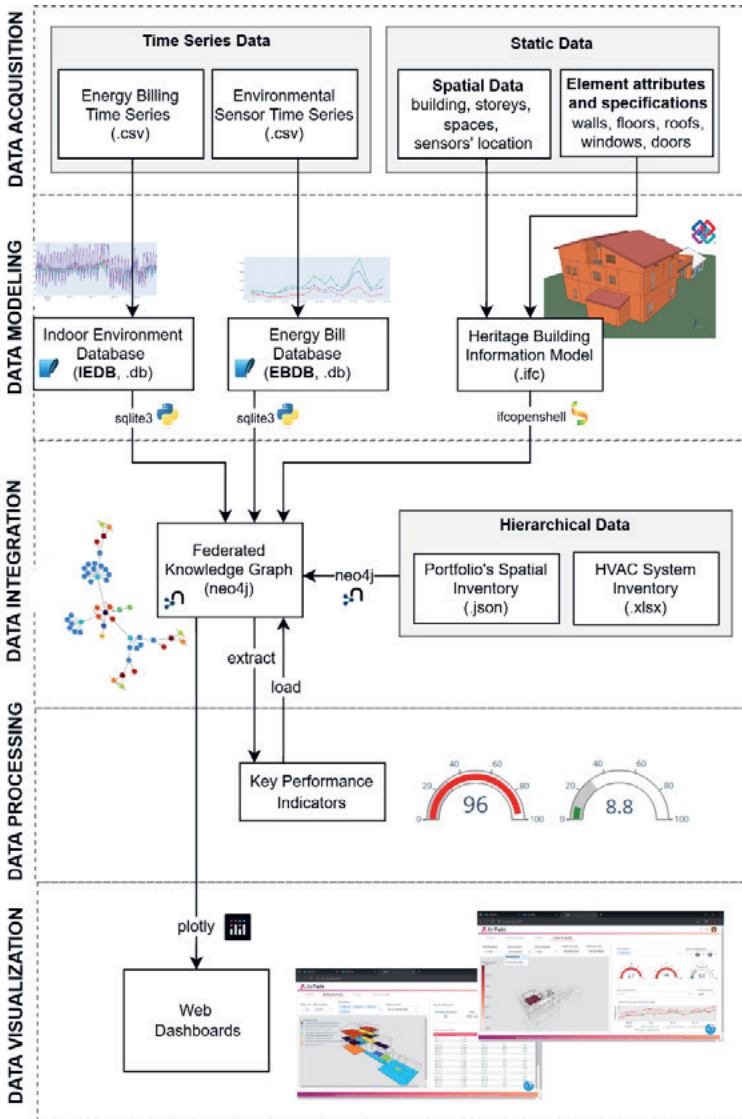


Figure 84. Methodological workflow for delivering the federated KG and the prototype DSS.

incorporated into the CAD floor plans with accompanying descriptive legends.

- *Material and construction data.* On-site investigations and archival analysis provided detailed information on the materials composing the building envelopes. In addition, thermofluximetric analyses were conducted to determine the thermal transmittance of the main walls. Where data were unavailable, informed assumptions were made based on typological considerations. The output of this phase was a set of construction element schedules in CAD format, covering both opaque and transparent envelope components.
- *Sensor measurements.* A dedicated sensor network recorded temperature and relative humidity at ten-minute intervals. Sensors were installed in representative interior rooms, near exterior walls, and in selected outdoor areas. The resulting time-series data were stored as comma-separated values (CSV) files.
- *Utility bills.* Monthly electricity invoices were digitised, providing consumption (kWh) and cost (EUR) data over several years. Electricity use covered heating, cooling, lighting, and appliances. The resulting data were likewise stored in CSV format.
- *HVAC system inventory.* Through on-site inspections and analysis of technical documentation, all mechanical systems' devices (including heating, cooling, ventilation, and lighting) were mapped and recorded. The compiled system inventory was organised in Excel spreadsheets (XLSX format).

The collected datasets were pre-processed in the data modelling phase to ensure their controlled ingestion into the graph environment. Geometric and spatial data, together with material and construction data, were modelled using HBIM (Heritage Building Information Modelling). This enabled the spatial and constructive characteristics of the buildings to be represented in a structured and semantically consistent manner. Time-series data, including sensor measurements and utility bills, were processed to identify and correct null or inconsistent values. Once cleaned, the datasets were stored in a SQL database to ensure compatibility with subsequent integration into

the knowledge system. The HVAC system inventory was reformatted into a new Excel file, structured to comply with the semantic conventions defined within the project, as described in the following sections.

7.3.3 Data modelling

Heritage Building Information Models

The HBIM models shown in Figure 85 were created in Autodesk Revit 2023 using geometric and alphanumeric information derived from the data collection phase.

The main aim of the HBIM process was to produce information models containing relevant data on the spatial layout of buildings, the material characteristics of envelope and partition components, and the positioning of sensors.

The information models were created at a Level of Development (LOD) 300, with simplified geometry to streamline the digitalisation process, then exported in IFC format to ensure compatibility with the presented workflow. Indeed, rather than produce a detailed geometric virtual reconstruction of the buildings, the HBIM objective here was to build a “skeleton” of elements to which performance data could be attached.

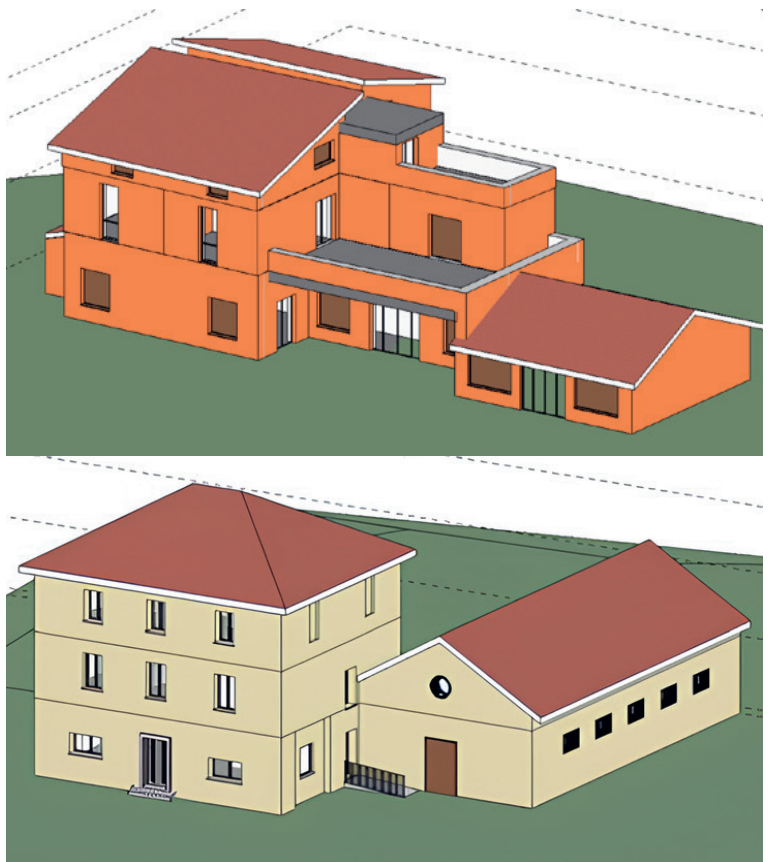
Spaces were exported as `IfcSpace` entities and enriched with a set of functional and environmental attributes, including a unique identifier (UID), net area and volume, designated function (classified according to OmniClass Table 13), occupancy count and requirements, thermal setpoints, and basic HVAC characteristics, namely whether the space is cooled, conditioned, or mechanically ventilated.

Opaque components, such as walls, floors, and roofs, were modelled in Revit using material layer sets and exported as `IfcWall` and `IfcSlab` entities. The materials comprising these layer sets were associated with thermophysical properties including density, thickness, thermal conductivity, and specific heat capacity. From these, aggregate parameters were derived for each opaque construction, including overall thickness, mass per unit area, thermal resistance, and thermal transmittance. A typology-based approach was adopted, supported by a structured schedule of construction types and a standardised naming convention to facilitate model reuse.

Doors and windows were modelled following the same typological logic, using custom Revit families and types explicitly developed for the case study, and exported as *IfcDoor* and *IfcWindow* entities. Each type was linked to essential thermal properties, including transmittance values for glazing, frame, and the complete unit, as well as frame material and glazing type. These data were derived from field inspections.

To enable integration with dynamic data, sensors were represented in the HBIM as placeholders, each assigned a UID corresponding to a KG entry, and exported as *IfcSensor* entities. These placeholders were linked to their respective spaces via containment relationships, enabling automated linkage to time-series sensor data during the integration phase.

Figure 85. HBIM models of the case studies in Autodesk Revit. At the top, the Capo di Bove complex. At the bottom, Villa dei Quintili. Source: (Coraglia, 2025)



Time-series databases

The billing records and sensor readings were stored in two dedicated SQLite databases: the Energy Bill Database (EBDB) and the Indoor Environment Database (IEDB).

The EBDB contains monthly electricity consumption (expressed in kWh) and the corresponding cost (in euros) for each building on each site, covering the period from January 2021 to December 2022. An excerpt from this database is presented in Table 12, showing the structure of the stored information.

The IEDB, by contrast, manages higher-resolution data. It includes temperature and relative humidity measurements collected in representative indoor spaces within the main buildings, along with data from one outdoor reference location for each site. Table 13 provides an excerpt illustrating the database structure. This repository follows the same organisational schema as the EBDB to ensure consistent representation of dynamic data. Referential integrity with the metadata stored in the HBIM is maintained by linking each record to its corresponding element via a unique identifier. Specifically, each observation in the EBDB is associated with an energy meter ID. At the same time, each record in the IEDB is linked to a sensor ID, thereby ensuring coherent cross-referencing between the static and dynamic data domains.

sosa:madeBySensor	sosa:ObservableProperty	Unit	Value	Timestamp
pt_CBC_EEU	ElectricEnergyUsage	kWh	4,552.00	2021-01-01
pt_CBC_EEU	ElectricEnergyUsage	kWh	4,156.00	2021-02-01
pt_CBC_EEU	ElectricEnergyUsage	kWh	4,242.00	2021-03-01
pt_CBC_EEU	ElectricEnergyUsage	kWh	3,424.00	2021-04-01

Table 12. Extract from the time-series database with bill recordings.

Table 13. Extract from the time-series database with sensor recordings.

sosa:madeBySensor	sosa:ObservableProperty	Unit	Value	Timestamp
pt_CBC-PT1_temp	Temperature	°C	22.00	24/5/23 10.00
pt_CBC-PT1_temp	Temperature	°C	22.04	24/5/23 10.10
pt_CBC-PT1_temp	Temperature	°C	22.07	24/5/23 10.20
pt_CBC-PT1_temp	Temperature	°C	22.09	24/5/23 10.30

HVAC system inventory

The HVAC systems were not modelled in the BIM environment due to their high complexity, frequent modifications, and the inherent difficulty of tracing all components within the heritage building fabric. Instead, these systems were documented in an external spreadsheet (XLSX format), which served as a detailed inventory subsequently integrated into the KG.

Although the geometric representation of HVAC components was not included in the HBIM, this approach enabled a structured description of equipment and connections via semantic relationships. The data structure supports future linkage to BIM objects using the exact referencing mechanism adopted for sensors, thereby ensuring the model's potential expandability.

An excerpt of the MEP dataset is presented in Table 14. Each row in the spreadsheet corresponds to an individual equipment item and includes information about its ontological class (specified in the 'type' column) and its associations with building systems and spatial entities.

7.3.4 Data integration

The data integration phase was articulated according to three main subphases, which are:

1. Ontological mapping and alignment;
2. Data serialization;
3. Generation of the KG.

Ontological mapping and alignment

Once the data had been modelled, semantic alignment was established between all the models and databases developed in

this study and the Federated Ontology Framework (FOF) introduced in Chapter 3. In particular, a subset of the FOF was adopted from BOT, Brick, and SOSA (Figure 86). At its core, the ontological framework was structured around a *spatial hierarchy*, defined as follows:

- `brick:Portfolio` represents the owner's portfolio, that is, the collection of buildings under its management;
- `brick:Region` defines the broader geographic area in which the portfolio is distributed, corresponding to the different areas of the park;
- `bot:Site` identifies a specific area containing one or more buildings;
- `bot:Building` denotes the individual building;
- `bot:Storey` refers to floor within buildings;
- `bot:Space` corresponds to room.

All these entities are semantically connected through the `brick:hasLocation` relationship.

In addition to the spatial hierarchy, a second structure, the *sensor hierarchy*, was introduced to model the dynamic data-acquisition system. According to this hierarchy:

- Sensors, understood as physical devices, are defined according to the `ifc:Sensor` class;
- The data points associated with each sensor are represented as `brick:Point`, and more specifically as instances of the subclasses `brick:Humidity_Sensor` and `brick:Temperature_Sensor`.
- Energy meters are defined as `brick:Meter`, more precisely, as `brick:Electrical_Meter`, and are linked to data points of the type `brick:Electric_Energy_Usage_Sensor`.

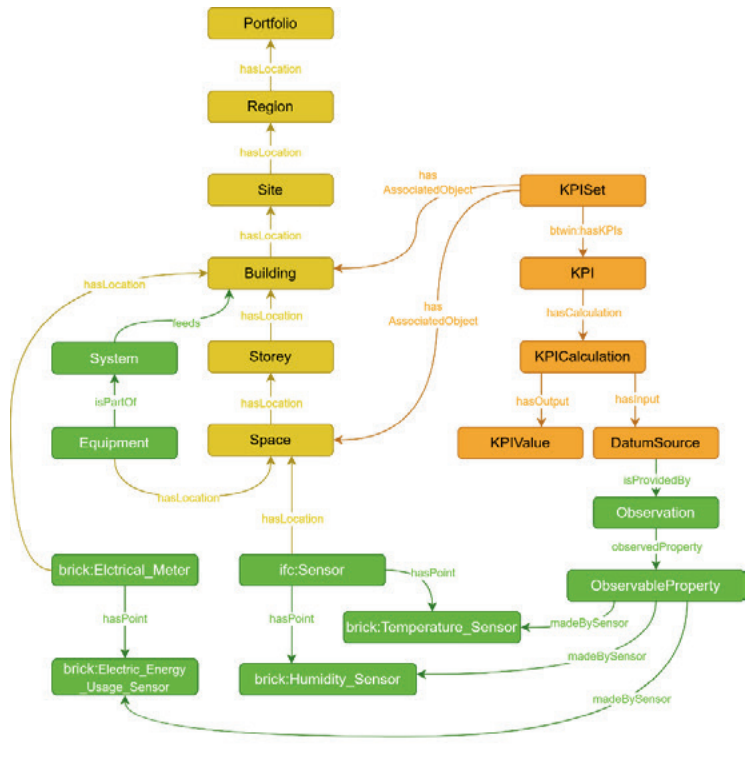
Each `ifc:Sensor` is connected to the space in which it is located through the `brick:hasLocation` relationship, while `brick:Point` instances are linked to their corresponding `ifc:Sensor` and `brick:Meter` entities through the `brick:isPointOf` relationship.

To this structure, the class `sosa:Observation` is added to represent individual data readings stored in the time-series databases. Each observation is connected to its corresponding `brick:Point` via the `sosa:madeBySensor` relationship, thereby establishing the link between the semantic graph and the dynamic measurement data.

id	name	type	brick:isPartOf System	brick:hasLocation Space
AHU001	AHU (below ceiling)	brick:Air_Handling_Unit	heatingSystem	CBC_PT.1
AHU002	AHU (below ceiling)	brick:Air_Handling_Unit	heatingSystem	CBC_PT.1
FC001	Fan coil	brick:Fan_Coil:Unit	heatingSystem	CBC_PT.3
HP001	Heat pump	brick:Packaged_Heat_Pump	heatingSystem	CBC_Ext
BL1.4-1	How water boiler	Brick:Electric_Boiler	dhwSystem	CBC_P1.4

Table 14. Extract from the HVAC inventory.

Figure 86. Subset of the FOF supporting the data integration process.



Legend

Equipment Module

Spatial Element Module

KPI Module

Equipment Module →

Spatial Element Module →

KPI Module →

The spatial and sensor hierarchies are complemented by the *system hierarchy*, which enables semantic mapping of the building service equipment in the HVAC inventory.

- `brick:System` is used to represent a collection of equipment and auxiliary components – such as control units, accessories, interconnecting elements, and terminal devices – through which energy is transformed to perform a specific function, for example, HVAC, domestic hot water production, or lighting.
- `brick:Equipment` represents individual devices that serve part or all of a building and may include systems for electric power, lighting, transport, or service water heating. Examples include furnaces, boilers, air conditioners, heat pumps, chillers, water heaters, luminaires, ballasts, elevators, and escalators, among others.

The containment relationship between `brick:Equipment` and `bot:Space` is expressed through the `brick:hasLocation` property. The membership relationship linking each `brick:Equipment` to its parent `brick:System` is defined by the `brick:isPartOf` property. Finally, the service relationship describing the connection between systems and the spatial entities they serve is represented by `brick:feeds`. The same `brick:feeds` relationship is also used to define dependencies between two pieces of equipment (for example, ‘a boiler that feeds a radiator’).

Finally, the *performance hierarchy* allows for the representation of performance-related data. Energy- and IE-related metrics are modelled as `eko:KPI` objects, which represent individual key performance indicators derived from the time-series data. KPIs are grouped into `btwin:KPISet` entities, each corresponding to a collection of related metrics computed for a specific spatial level. For instance, building-level KPI sets aggregated indicators such as total energy use, cost, and CO₂ emissions. In contrast, space-level KPI sets included metrics such as average temperature, relative humidity, and comfort indices. The `eko:hasAssociatedObject` relationship connects each `btwin:KPISet` to its respective spatial entity: `bot:Building` for energy-related indicators and `bot:Space` for IE-related indicators. This semantic linkage ensured that every performance metric could be directly contextualised within its physical environment, enabling seamless queries across geometric and performance domains.

Data serialisation

To enable seamless data linking, data was serialised according to the FOF using the JSON-LD presented in Section 4.3.

To translate native data into the JSON-LD schema, a suite of BTwin's connectors was used, each transforming domain-specific constructs into ontology-compliant JSON entities. For example, the IFC Connector, built on IfcOpenShell, parsed IFC files to extract buildings, storeys, spaces, and their properties, and emits the corresponding JSON-LD graph objects. The SQLite connector, in contrast, enabled the mapping of sensor points and SOSA observations to their corresponding sensor devices and the spaces that host them.

This set of connectors and functions automates the ETL process, ensuring that every digital artefact – whether geometric, spatial, temporal, or performance-related – is consistently represented.

Generating the knowledge graph

Using the BTwin toolkit, a Neo4j Labelled Property Graph (LPG) was developed by integrating static and dynamic datasets through the hybrid architecture outlined in Chapter 3.

At the highest level, the spatial hierarchy was manually initiated by creating `brick:Region` and `bot:Site` nodes, which were then connected to lower-level spatial structures. These subordinate hierarchies were generated automatically via the IFC connector, which extracted `ifcBuilding`, `ifcBuildingStorey`, and `ifcSpace` entities from the buildings' IFC models. Containment relationships (`brick:hasLocation`) among regions, sites, and buildings were manually defined, while those between buildings, storeys, and spaces were directly inherited from the IFC structure.

The IFC connector also enabled the creation of `ifc:Sensor` nodes, whose spatial containment in `bot:Space` nodes was refined using Topologicpy containment functions. Separate `brick:Point` entities were generated to represent individual measurement channels (i.e., temperature, humidity, electricity consumption, and cost), which were subsequently mapped to the appropriate Brick subclasses.

Additional `btwin:Document` nodes were created for each key file associated with each building, including the IFC, EBDB,

IEDB, and system inventory. Each document was uniquely identified by its file path and linked to both the building it described and the corresponding time-series or equipment data.

The SQLite connector was used to process dynamic databases and to aggregate data at either the space or building level. Connections between measurement points and `sosa:Observation` instances were established through the `sosa:madeBySensor` field within the databases. The SQLite data were imported into Python and processed using pandas dataframes, with values aggregated over the monitoring period for each building or space. The processed results were modelled as `eko:KPI` objects, grouped within `btwin:KPISet` nodes, each linked to its corresponding spatial entity. KPI values encompassed building-level indicators, such as total energy use, energy cost, and associated CO₂ emissions, as well as environmental monitoring metrics, including average temperature, average humidity, and maximum temperature.

Finally, the HVAC system inventory was imported into Python and converted from Excel to the Neo4j environment. To achieve this, the necessary `brick:System` nodes were first created to represent the systems installed in the case study buildings. For each inventory row, a corresponding node was generated in the graph and labelled according to its type. Each node was linked to its parent system through the `brick:isPartOf` relationship and to the space it occupies through the `brick:hasLocation` relationship.

The resulting KG, depicted in Figure 87 and Figure 88, enabled multi-level querying of energy and indoor environmental metrics, allowing information retrieval at varying levels of granularity, from the entire building down to individual spaces. For example, total energy consumption for a building during the monitoring period can be retrieved using the Cypher query shown in Figure 89, while the average temperature of a specific space can be obtained using the query shown in Figure 90.


```

MATCH (building:bot_Building {UID: 'building01'})
MATCH (building)<-[:eko_hasAssociatedObject]-(:kpi:btwin_KPISet)
WHERE kpi.UID CONTAINS 'energy'
RETURN building.UID as BUILDING,
       kpi.UID as KPI,
       kpi as KPI_DATA

```

Figure 89. Cypher query performed on Neo4j to retrieve energy consumption KPIs for a building.

```

MATCH (building:bot_Building {UID: 'building01'})
MATCH (building)<-[:brick_hasLocation*]-(:space:bot_Space)
MATCH (space)<-[:eko_hasAssociatedObject]-(:spacekpi:btwin_KPISet)
WHERE spacekpi.UID CONTAINS 'temperature'
RETURN building.UID as BUILDING,
       space.UID as SPACE,
       spacekpi.UID as KPI,
       spacekpi as KPI_DATA

```

Figure 90. Cypher query performed on Neo4j to retrieve temperature-related KPIs for the building's spaces.

7.3.5 Data processing

At this stage of the workflow, the KG and the complementary time-series databases were jointly queried to derive meaningful indicators of building performance.

Specifically, the EBDB was interrogated to compute three primary KPIs at the whole-building level. These are:

- *Energy Usage* (EU): Calculated by summing all interval readings over the selected time period;
- *Energy Cost* (EC): Obtained by applying the corresponding tariff rate to the total energy consumption derived from the billing records;
- *Equivalent CO₂ Emissions* (CO₂E): Estimated by multiplying the energy usage by an appropriate emissions factor.

These KPIs can be aggregated over monthly, yearly, or custom-defined intervals and normalised to the building's net floor area or volume to enable comparative analysis. In parallel, the IEDB was queried for processing microclimatic data, including indoor and outdoor temperature and relative humidity, obtained through sensor measurements. These data were aggregated at the room level with an hourly resolution, enabling the calculation of descriptive statistics, such as average, minimum, and maximum temperature over the monitoring period or more specific subperiods².

After computation, all KPIs were instantiated as nodes within the KG and classified under the `eko:KPI` class, grouped under

btwin:KPISet nodes (Figure 91).

Each KPISet was then linked to the relevant spatial entity, bot:Building or bot:Space, via the eko:hasAssociatedObject relationship. This structure mirrors the underlying graph hierarchy, which encodes spatial, building-to-meter, and sensor-to-space relationships. The resulting integration allows seamless navigation between high-level building metrics and individual meter or sensor observations, thereby supporting multi-scale performance analysis.

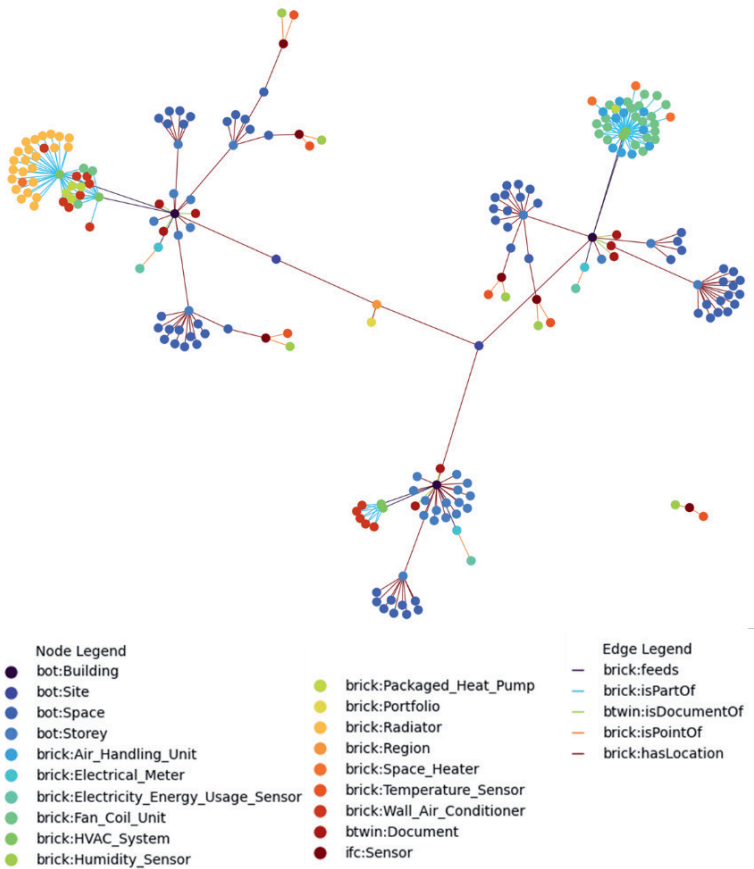


Figure 91. Complete KG for the two building complexes analysed.

7.4 Results and outcomes

The results of this process were made available through a web-based application designed to support data visualisation and enable users to perform customised queries to explore and analyse the building information.

Since interaction with the Neo4j graph database is not restricted to users with expertise in graph technologies, the web application was developed as a set of interactive dashboards, providing a more intuitive interface for data exploration and visualisation and thus allowing facility managers to visualise data (Figure 92). Once data is linked in the KG, the app enables it to be extracted via custom query functions built on Neo4j APIs and displayed in a series of charts and tables. The dashboards, developed according to Business Intelligence (BI) principles, include:

- A section where the model's geometry can be viewed in 3D, with spaces colour-coded according to their properties or KPIs;
- A section displaying the building's KPIs, via gauge charts and other types of visualisations;
- A section showing time-series plots for more in-depth analysis of the raw data fed into the platform. The plot updates automatically in response to filter adjustments and supports aggregation at yearly, monthly, daily, or hourly resolutions, with statistical operators including minimum, maximum, mean, and sum.

Each of these sections is equipped with dropdown menus and other controls to filter the data by user request: e.g., selecting a specific room or floor, choosing the property used to colour the geometry, picking which KPIs to display, or applying temporal and topological filters to the time-series.

Even in a simplified form, this visualisation approach proved accessible to non-technical users, who provided feedback and support in participatory processes to improve it.

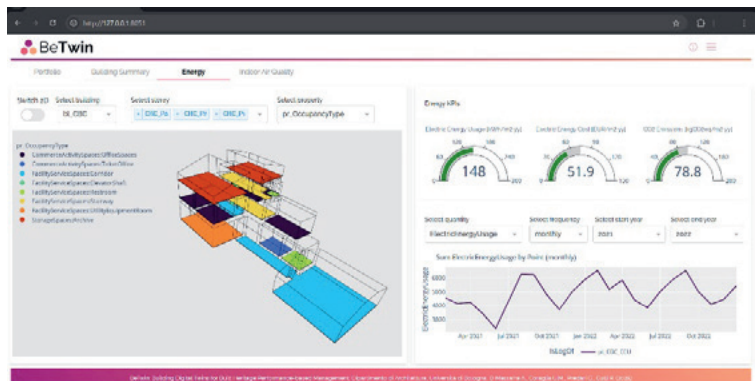
Overall, this application demonstrated the methodological flexibility of using semantic graphs for data integration in the characterisation process of heritage buildings. By connecting information on building topology, construction properties, HVAC systems, and sensor data, the resulting KG established

a holistic knowledge system capable of representing both the physical and performance characteristics of the heritage asset.

The hybrid architecture, combining a KG with time-series databases, demonstrated that static and dynamic data can co-exist effectively within the same semantic environment without overloading the graph. This approach aligns with broader digitalisation trends in heritage management. It is particularly well suited for extension into a Digital Twin (DT) framework, in which static and dynamic data will be continuously integrated and bidirectionally exchanged with the physical asset to optimise its operational performance.

Graph networks offered several advantages over traditional data structures. First, they made relationships explicit and navigable, allowing complex queries that traverse the building's spatial hierarchy while linking it to sensor and performance data. Second, they proved inherently extensible, as new classes and properties can be incorporated without disrupting existing datasets, thereby accommodating new types of information as the building's life cycle evolves. Finally, the ability to query and visualise data across multiple scales fostered a deeper understanding of the interplay between fabric, environment, and use, a key requirement for planning data-informed renovation strategies. In the next chapter, the value of graph-based approaches for data integration will be further extended to address model interoperability, demonstrating how the KG can interlink a Building Information Model (BIM) with a Building Energy Model (BEM) and high-resolution simulation data.

Figure 92. Prototype dashboard to access and visualise data about the main building of the Capo di Bove complex.



Notes

¹ For instance, when examining the building envelope, researchers point to significant uncertainties in determining thermophysical properties, largely due to factors such as moisture presence and the heterogeneity of traditional materials (Al-Addous and Albatayneh 2020). The development of non-destructive methods, such as infrared thermography (Patel et al. 2023; Takva et al. 2025), led to the spread of tools capable of providing the critical parameters needed for energy diagnostics; however, these are not the only data required for performance characterisation. In addition to physical measurements, examining historical records, archival sources, and energy bills, and carrying out geometric surveys and monitoring occupancy patterns and environmental conditions (e.g., via sensors) are essential for developing comprehensive knowledge bases.

² As the research progresses, this repository will form the foundation of a more comprehensive database, designed to support thermal comfort and environmental performance assessments.

8. PERFORMANCE GRAPHS FOR DECISION SUPPORT IN ENERGY MANAGEMENT

Abstract

As seen in the previous chapter, KGs serve as a powerful integration layer in modern digital services. This is not only for linking different data types but also for connecting diverse models.

This section further advances the idea of graph-driven system interoperability by demonstrating how KGs can effectively bridge BIM and Building Energy Models (BEMs). Built on previous research published in the article *'Building Information Modelling and Building Performance Simulation-Based Decision Support Systems for Improved Built Heritage Operation'* (Massafra *et al.* 2023), the text presents a graph-based workflow that integrates Industry Foundation Classes (IFC) models with EnergyPlus (EP) simulation to form a knowledge system that enables fine-grained, cross-domain analysis about energy needs in university buildings.

The case study is the School of Engineering and Architecture in Bologna, a rationalist, listed building designed in the 1930s by the architect Giuseppe Vaccaro. This research was initially developed during my doctoral studies at the Department of Architecture, University of Bologna (2020–2024), and was subsequently continued within the ongoing Horizon Europe project HERIT4AGES (Grant Agreement No. 101123175), where the building is serving as a demonstration site for testing the project's Digital Twin Ecosystem. Special thanks go to Prof. Riccardo Gulli for his guidance throughout this work, as well as to my colleagues and friends who, over the past years, have worked on this building, contributing to the creation of a highly detailed data corpus that supported this research.

8.1 Contextualisation and use case definition

Continuing the discussion within the ‘KGs for data integration’ thread, this chapter focuses on the use of graphs for managing high-granularity performance data coming from simulation models. Unlike the preceding example, which, among other aspects, explored the linkage between BIM and sensor observations, this case introduces an additional layer of complexity by integrating a BIM with Building Performance Simulation (BPS) models¹, specifically BEM. The aim is to demonstrate that KGs can mediate not only between heterogeneous datasets but also between digital models based on distinct data schemas, thereby forming interoperable knowledge systems to support performance-oriented decision-making, for instance, in energy management.

More specifically, the research presented in the following text aims to prototype a digital workflow that connects a building IFC model, developed through the *Topological BIM* (TBIM) procedure², with a BEM directly derived from it. The interaction between the two models, coupled with energy simulation runs, enables the calculation of KPIs relevant to energy management and the integration of all related data into a unified knowledge system.

This system is made accessible via interactive dashboards that visualise relevant data through graphical charts, offering energy managers actionable insights, such as energy demand per space, related operational costs, and carbon emissions. This information apparatus enables comparative analyses across different spaces (helping determine intervention priorities) and the detection of mismatches between planned occupancy and resulting energy loads (thereby enabling adjustments to occupancy schedules and system operation strategies).

In the workflow, KGs are mainly employed to resolve BIM-to-BEM and BIM-to-BES interoperability challenges. This diverges from traditional file-based exchange mechanisms between BIM and BEM (such as IFC-to-gbXML), which have long been examined in the literature and are known to introduce data duplication and require repeated manual updates. KGs enable both models to coexist within a unified environment. Each model remains in its native data format, retains its specialised infor-

mation, and allows data to be retrieved directly from the shared ecosystem as emerging needs arise. A further implication of this requirement is the integration not only of the models themselves but also of the dynamic simulation outputs generated by EP. This is achieved through a hybrid architecture that pairs the KG, which represents static relationships, with a time-series database that directly stores time-series simulation data.

8.2 Case study: energy management of public assets

The case study selected for testing the research comes from the Italian university building stock, a representative subset of assets owned by local public administrations. University buildings present, on the one hand, a distinctive functional diversity that leads to complex interactions between users and energy needs. On the other hand, they often exhibit architectural and cultural value that must be safeguarded, meaning that, when physical interventions are limited, energy-efficiency measures must rely on operational and management strategies.

The management of these facilities is further influenced by the structure of public administrations, which, in their various and often complex forms, face systemic constraints that limit their ability to implement performance-driven, cost-optimal improvements. This limited capacity, combined with the absence of adequate decision-support tools, frequently results in inefficient use of public resources, making it difficult to assess how assets are being utilised and to plan corrective actions aligned with strategic objectives.

Within this broader context, the research examines the energy-related challenges affecting higher education buildings during their operational phase. This focus is increasingly significant, as in recent years universities worldwide have adopted ambitious targets for energy reduction and emissions control, prompting the introduction of numerous measures aimed at lowering energy consumption ('Massachusetts Institute of Technology, From Plan to Action: MIT Campus Greenhouse Gas Emissions Reduction Strategy, 2017.' n.d.; 'Queen's University, Queen's University Greenhouse Gas Inventory Report 2017, 2018.' n.d.; 'University of West England, Carbon Management

Plan 2013-2020 (Version: 03), 2017.' n.d.), including the installation of renewable energy systems, the refurbishment of ageing facilities, and the introduction of policies aimed at raising user awareness and encouraging more energy-conscious behaviours.

In addition to these measures, other valuable (yet often overlooked) strategies can significantly benefit university buildings, such as flexible working arrangements and demand-driven control of building systems³ (Mosteiro-Romero *et al.* 2023). Tools capable of revealing the strategic impacts of such interventions are increasingly essential for determining which actions to fund and for optimising the allocation of technical and economic resources across large building portfolios.

More specifically, the University of Bologna's building stock was selected as the portfolio-scale case study⁴. Within it, a representative building-scale case study was identified as the School of Engineering and Architecture in Bologna⁵ (Figure 93).

Like many historic buildings, the facility has intrinsic features that limit its potential for energy improvements. This became particularly evident during the COVID-19 pandemic, when the building, despite remaining unoccupied for several months, still showed above-average heating consumption⁶. Its substantial energy use, combined with highly variable occupancy patterns that peak during teaching hours and drop sharply during holiday periods, made it an ideal testbed for analysing the relationship between occupancy and energy demand. Because the building was not equipped with sensors or Internet of Things (IoT) devices – a condition typical of many public-sector buildings – the study relied on BPS data, which were calibrated against utility bills as documented below.

Figure 93. Aerial photography of the School of Engineering and Architecture of the University of Bologna.



8.3 Methods and tools

8.3.1 Workflow articulation

The methodology adopted to pursue the research objectives is organised into three main phases, described in the following sections:

1. Data acquisition;
2. Data modelling;
3. Data processing;
4. Data integration;
5. Data analysis and visualisation.

8.3.2 Data acquisition

During the data acquisition phase, a comprehensive dataset was collected to enable the development of an accurate dynamic energy model in EP. The dataset includes the following categories:

- *Historical information*: archival and bibliographic research was carried out to reconstruct the building's construction history, architectural characteristics, and subsequent transformations from its original configuration to the present day. These analyses build upon prior studies conducted over the last years by the Department of Architecture, University of Bologna, documented in previous publications (Predari *et al.* 2022).

- *Construction information*: material and construction data collected across previous research projects on the same building were consolidated. These include envelope and structural material properties as well as typological catalogues of recurring construction elements.
- *Functional layout*: architectural CAD drawings, digital surveys, and space inventories were obtained from the building management office. These datasets were required to define the building's functional organisation and to support subsequent occupancy modelling activities.
- *HVAC systems*: information on the mechanical systems serving the building (heated and cooled zones, ventilation strategies, and control logic) was gathered through documentation provided by building managers and through on-site discussions. Operational schedules and system settings were also acquired to support simulation inputs.
- *Energy bills*: electricity and thermal energy bills for the period 2021–2023 were collected to calibrate the simulation model against measured consumption data.
- *Occupancy data*: occupancy schedules for all major spaces were derived from the university's asset management and room booking systems. These records were combined with data from the institutional learning platform to estimate hourly attendance for each course throughout the academic year. For office spaces, typical occupancy profiles were defined based on standard building opening hours.

8.3.3 Data modelling

Once acquired, all datasets were structured and integrated within two digital models, which served as the primary reference sources for subsequent analyses: a BIM in IFC format and a BEM in EP's Input Data Format (IDF). The BIM model was intended to provide a semantic and geometric framework for linking all information. In contrast, the BEM was intended to enable dynamic energy simulations and the computation of performance indicators. An additional data source consists of occupancy schedules, modelled in JSON format compliant with the IDF to capture planned occupancy conditions at an hourly resolution.

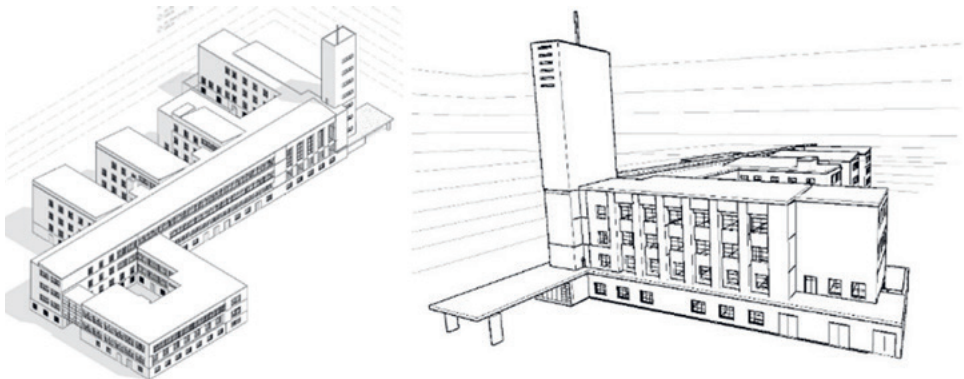
Building Information Modelling

The first model generated from the collected information was the BIM model. It was created using the TBIM methodology, ensuring that all information required for the energy analysis was included.

The BIM model (Figure 94) was developed in Autodesk Revit and then exported in IFC to include the following spatial elements: spaces (IfcSpace), zones (IfcZone), and floors (Ifc-BuildingStorey), as well as the following construction elements: walls (IfcWall), slabs and roofs (IfcSlab), windows (IfcWindow), and doors (IfcDoor).

The occupancy-, thermo-hygrometric-, and energy-related properties were semi-automatically assigned to spaces following the TBIM procedure, which consists of mapping each space's function and assigning the corresponding attributes to its functional category (see Table 15). Subsequently, using the same semi-automated process, thermo-physical and material properties were assigned to the opaque envelope components (walls, roofs, and floors) and to the transparent components (doors and windows) (see Table 16 and Table 17). Minor manual adjustments were made to specific object instances within Revit before exporting the model to IFC.

Figure 94. BIM model of the School of Engineering and Architecture in Bologna. Axonometric view, on the left. Perspective view, on the right.



Property Name	Description	Quantity	Unit
Artificial Lighting	Indicates whether this space requires artificial lighting (as natural lighting would not be sufficient). (TRUE) indicates yes (FALSE) otherwise.	Boolean	-
Equipment Power Density	The maximum electrical power input (in Watts/m ²) to using electric appliances in a zone, including PCs, plotters, elevators and other equipment, if present.	Power Density	W/m ²
Illuminance	Required average illuminance value for this space.	Illuminance	lux
Is Cooled	Indicates whether this space requires air conditioning (TRUE) or not (FALSE).	Boolean	bool
Is Heated	Indicates whether this space requires heating provided (TRUE) or not (FALSE).	Boolean	bool
Is Mechanically Ventilated	Indicates whether the space is required to have mechanical ventilation (TRUE) or not (FALSE).	Boolean	bool
Is Naturally Ventilated	Indicates whether the space is required to have natural ventilation (TRUE) or not (FALSE).	Boolean	bool
Is Occupied	Indicates whether the space is permanently occupied (TRUE) or not (FALSE).	Boolean	-
Lighting Level	The maximum electrical power input (in Watts) to lighting in a zone, including ballasts, if present.	Power	W
Lighting Power Density	The maximum electrical power input (in Watts/m ²) to lighting in a zone, including ballasts, if present.	Power Density	W/m ²
Mechanical Ventilation Rate	Indication of the requirement of a particular mechanical air ventilation rate, given in air changes per hour.	Air Changes	h ⁻¹
Mechanical Ventilation Rate Per Person	<i>Mechanical Ventilation Rate Per Person</i> standard value for mechanical ventilation of spaces given by UNI EN 10339.	Air Changes	L/s pp

Table 15. Space properties within the BIM model.

Property Name	Description	Quantity	Unit
Mechanical Ventilation Rate Per Volume	<i>Mechanical Ventilation Rate</i> standard value for mechanical ventilation of special spaces (e.g. toilets) given by UNI EN 10339.	Air Changes	h ⁻¹
Natural Ventilation Rate	Indication of the requirement of a particular natural air ventilation rate, given in air changes per hour.	Air Changes	h ⁻¹
Occupancy Density Operation	Design occupancy loading for this type of usage assigned to this space according to UNI EN 10339.	Occupancy Density	pp/m ²
Occupancy Number	Number of people required for the activity assigned to this space.	People Count	pp
Occupancy Type	Occupancy type for the space object.	Text	-
Space Humidity Max	Max humidity of the space or zone that is required from the user/designer viewpoint	Relative Humidity	%
Space Humidity Min	Minimum humidity of the space or zone that is required from the user/designer viewpoint	Relative Humidity	%
Space Temperature Summer Max	Maximum temperature of the space or zone during the hot (summer) period, as required from the user/designer's viewpoint.	Temperature	°C
Space Temperature Summer Min	Minimum temperature of the space or zone for the hot (summer) period, as required from the user/designer's viewpoint.	Temperature	°C
Space Temperature Winter Max	Maximum temperature of the space or zone for the cold (winter) period, as required from the user/designer's viewpoint.	Temperature	°C
Space Temperature Winter Min	Minimum temperature of the space or zone for the cold (winter) period, as required from the user/designer's viewpoint.	Temperature	°C
Space Wind Load Rating	Wind load resistance rating for the windows and doors located in the space. It is provided in accordance with the national building code.	Text	-

Property Name	Description	Quantity	Unit
Construction Name	Name of the set of materials associated with walls, roofs, and floors, within the BIM model	IfcLabel	-
Thermal Transmittance	Thermal transmittance coefficient (U-Value) of an element.	IfcThermalTransmittanceMeasure	W/m ² K

Table 16: Thermal properties automatically added to the interface elements of the BIM model (walls, slabs, and roofs).

Property Name	Description	Quantity	Unit
Frame Material	Material of the frame of the opening	IfcLabel	-
Glass Layers	Number of glass layers within the frame. E.g. "2" for double glazing.	IfcCountMeasure"	-
Glass Thermal Transmittance	Thermal transmittance coefficient (U-Value) of the gas.	IfcThermalTransmittanceMeasure	W/m ² K
Glazing Area Fraction	The fraction of the glazing area relative to the total area of the filling element.	IfcNormalisedRatioMeasure"	-
Height	Total outer height of the window lining	IfcPositiveLengthMeasure"	m
Solar Heat Gain Transmittance	The ratio of incident solar radiation that contributes to the heat gain of the interior.	IfcNormalisedRatioMeasure	.
Thermal Transmittance	Thermal transmittance coefficient (U-Value) of the window.	IfcThermalTransmittanceMeasure	W/m ² K
Visible Light Transmittance	The fraction of the visible light that passes through the object at normal incidence. It is a value without a unit.	IfcNormalisedRatioMeasure"	-
Width	Total outer width of the window lining.	IfcPositiveLengthMeasure"	m

Table 17: Properties manually added to the apertures of the BIM model (windows and doors)

Building Energy Model

After generating the BIM, the focus shifted to developing the BEM. To create the BEM, an IDF model of the building was produced from the previously described BIM model, following the procedure detailed in an earlier publication (Massafra and Gulli 2023). This workflow employs Visual Programming tools (Grasshopper, Rhino.Inside, and Ladybug Tools) as intermediaries between BIM and BEM.

The resulting model included all the data needed to run EP simulations. This information covered zone-related parameters, such as occupancy (area per occupant, maximum number of occupants, occupancy types, and occupancy schedules), geometric characteristics (area, volume, height, and shape), and space conditioning and lighting requirements (temperature and lighting setpoints, lighting power density, equipment power density, lighting schedules, ventilation rates, and HVAC schedules). It also incorporated the material and thermal properties of opaque surfaces (material layers, conductivity, thermal transmittance, and volumetric heat capacity) and transparent surfaces (U-value, solar heat gain coefficient, and visible transmittance).

Occupancy schedules

The HVAC and occupancy schedules were generated using a Python script, supported by Ladybug Tools, and exported as CSV files⁷. These schedules were then linked to the BIM and, consequently, to the BEM through UIDs corresponding to zone names (Figure 96).

Simulation time-series

After linking the schedules to the BEM, the simulations were run.

To prepare for the energy simulation in EP, the simulation settings were configured using components from Ladybug Tools in Grasshopper. These settings included the simulation timestep, simulation period, selected output variables, shadow calculation method, and other relevant parameters.

The energy model was calibrated by comparing the average actual annual energy consumption reported on energy bills from 2021 to 2023 with the simulated results for the entire building⁸.

The simulation output consisted of a CSV file containing hourly data for each building zone. Each data point represented the value of a specific variable for a particular hour of the year and a specific zone. In total, EP produced 3,723,000 observations, corresponding to 85 zones \times 8,760 hours \times 5 quantities (heating energy demand, cooling energy demand, lighting energy demand, equipment energy demand, and occupant count).

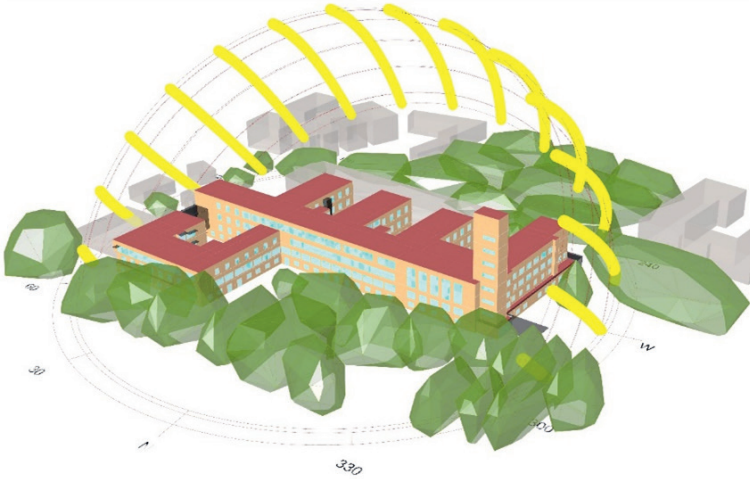


Figure 95. BEM model of the School of Engineering and Architecture in Bologna.

Proprietà	
Locali (1)	
Fasi	
Fase	
Analisi energetica	
dtu_HBZone	Aula0.4
dtu_HBZone_bldgProgram	SecondarySchool:Classro...
dtu_HBZone_numero	ZN_PO-A0.01.01
dtu_HBZone_Schedule_Occupancy	dtu_occSchedule_School
dtu_HBZone_Schedule_Activity	dtu_actSchedule_School
dtu_HBZone_IsConditioned	True
dtu_OccupancySchedule_ID	SchOcc_Aula0.4
dtu_ActivitySchedule_ID	Sch-Act-Offices
dtu_EquipmentLoadPerArea	2.000000
e.ZN_Zone_ID	ZN_PO-A0.01.01
Parametri IFC	
ifcExportAs	IfcSpace
p.ZN_AirConditioning	<input checked="" type="checkbox"/>
p.ZN_AirConditioningCentral	<input checked="" type="checkbox"/>
p.ZN_DiscontinuedHeating	<input checked="" type="checkbox"/>

Figure 96. At the top, the Occupancy Schedule ID is associated with Revit's Spaces. At the bottom, the schedule CSV file.

Honeybee Schedule file (to be used in combination with a thermal simulation program), , , Continuous Schedule file address:c:\ladybug\EPCSVSchedules\SCHSch-Occ_Aula0.4.csv; SCHSch-Occ_Aula0.4.csv: Thi 1,, ,Values [units],N/A month, day, hour, values

```

0, 1, 0.5, ,0.0
0, 1, 1.5, ,0.0
0, 1, 2.5, ,0.0
0, 1, 3.5, ,0.0
0, 1, 4.5, ,0.0
0, 1, 5.5, ,0.0
0, 1, 6.5, ,0.0
0, 1, 7.5, ,0.0
0, 1, 8.5, ,0.0
0, 1, 9.5, ,0.0
0, 1, 10.5, ,0.0
0, 1, 11.5, ,0.0
0, 1, 12.5, ,0.0
0, 1, 13.5, ,0.0
0, 1, 14.5, ,0.0

```

8.3.4 Data processing

The simulation data were used to calculate specific energy-related KPIs for each zone in the building. To do this, a Python script read the hourly EP output variables for each zone, converted them to the required units, and aggregated them to yearly values. These yearly values were then used to produce KPIs at two levels: per zone and for the entire building.

Four KPI sets were defined: *Energy*, *Cost*, *Environment*, and *Occupancy*. Across these sets, five KPIs were included: *Natural Gas Demand* and *Electricity Demand*⁹ (Energy KPI set), *Energy Cost*¹⁰ (Cost KPI set), *Equivalent CO₂ Emissions*¹¹ (Environment KPI set), and *Occupancy Rate*¹² (Occupancy KPI set). Table 18 summarises the KPIs and provides their definitions.

Table 18. KPIs calculated from simulations for each building zone.

KPI Name	Quantity	Unit	KPI Set Name	Description
Natural Gas Demand	Natural gas volume	smc/m ²	Energy	The volume of natural gas required for heating the zone.
Electricity Demand	Energy	kWh/m ²	Energy	The amount of electrical energy needed to operate the space.
CO₂ Emission	CO ₂ equivalent	kgCO ₂ eq/m ²	Environment	The CO ₂ -equivalent emissions resulting from energy consumption.
Energy Cost	Expenses	EUR/m ²	Cost	The cost associated with energy consumption.
Occupancy Rate	Occupancy density	pp	Occupancy	The average number of people within the room during occupation time

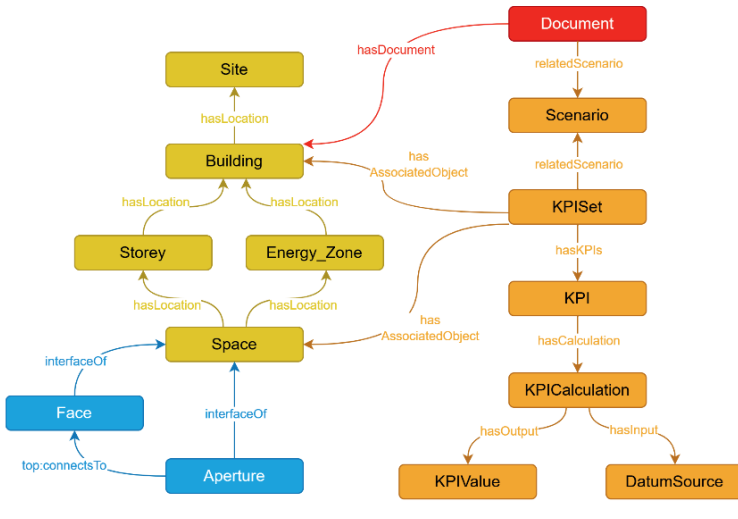
8.3.5 Data integration

Semantic alignment

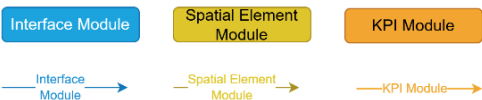
At the end of the data processing phase, the following data were available: the BIM, the BEM, and the simulation data (in the EP output CSV format).

According to the Federated Ontology Framework (FOF) presented in Chapter 3, semantic alignment was established between the classes, objects and information contained in the models and the time series file. In particular, a subset of ontology concepts was adopted, particularly from BOT and the Brick Schema, as presented in Figure 97. The corresponding semantic attributions and mappings are reported below.

Figure 97. The subset of the FOF used for mapping the case study domain.



Legend



The *spatial hierarchy* supporting this application is defined as follows:

- bot:Site identifies the area containing the building;
- bot:Building denotes the individual building;
- bot:Storey refers to a floor within the building;
- brick:Energy_Zone defines groups of spaces forming thermally uniform zones, as required for energy modelling;
- bot:Space corresponds to individual rooms.

All these entities are semantically connected through the brick:hasLocation relationship, which defines their hierarchical and spatial interdependencies.

The interface elements, namely those elements that delimit spaces, are modelled as follows:

- top:Interface: includes walls, slabs, and roofs;
- top:Aperture: includes windows and doors.

The top:Interface entities are connected to bot:Space through the bot:isInterface relationship, while top:Aperture entities are linked to top:Interface via the top:connectsTo relationship.

Finally, the *performance hierarchy* provides a structured representation of all performance-related information. Energy metrics are defined as eko:KPI objects, each representing a specific key performance indicator derived from time-series data. These KPIs are aggregated into btwin:KPISet entities, which group related indicators computed for each brick:Energy_Zone. The relationship eko:hasAssociatedObject links every btwin:KPISet to its corresponding zone and to the building, ensuring that each performance metric is contextually anchored within a spatial element.

Data serialisation

To enable seamless data integration, all information was serialised in accordance with the FOF using JSON-LD format. The translation of native data into this ontology-compliant schema was achieved through a suite of BTwin connectors, each designed to transform domain-specific constructs into JSON entities aligned with the ontology.

The IFC connector, built on IfcOpenShell, parsed IFC files to extract buildings, storeys, spaces, and their associated properties, and subsequently generated the corresponding JSON-LD

graph objects. Conversely, the IDF connector enabled the mapping of `brick:Energy_Zone` entities. Table 19 summarises the class alignment performed between the FOF and the BIM and BEM models.

In addition, another connector was implemented to process the outputs of the energy simulations (CSV file), converting them into entities compliant with `eko:KPI` and `btwin:KPISet` classes. Each KPI set was serialised in accordance with the previously described semantic rules and linked to its corresponding zone.

Generating the knowledge graph

Using the BTwin toolkit, a Neo4j-labelled property graph (LPG) was developed by integrating static information from the models with dynamic datasets derived from the simulations.

At the highest level, the spatial hierarchy was automatically generated by the IFC connector, which extracted the `ifcBuilding`, `ifcBuildingStorey`, and `ifcSpace` entities from the building's IFC model, along with their relationships. Conversely, the IDF connector enabled the creation of nodes representing thermal zones, corresponding to Zone elements in EP¹³.

Additional `btwin:Document` nodes were generated for each key file associated with every building, including the IFC, IDF, and EP output file. Each document was uniquely identified by its file path and linked to the corresponding building it described and to a fictional `kpi:Scenario`, representing the current scenario for the building.

The EP connector was then employed to process simulation outputs under dynamic conditions, aggregating results at the zone level and deriving project-specific KPIs.

The resulting LPG enabled multi-level querying of energy and occupancy metrics, supporting information retrieval at multiple granularities – from the entire building down to individual thermal zones. For example, the spatial hierarchy can be navigated to determine which spaces are contained within zones (Figure 12), or the energy needs of a specific zone can be obtained through the Cypher query shown in Figure 99.

Below are some representative figures depicting the resulting KG (from Figure 100 to Figure 103).

Table 19. Alignment between the FOF and the BIM and BEM classes.

FOF	BIM (Revit)	BIM (IFC)	BEM (IDF)
Bot:Building	Building	IfcBuilding	Building
Bot:Storey	Level	IfcBuildingStorey	-
Bot:Space	Space	IfcSpace	-
Brick:Energy:Zone	Zone	IfcZone	Zone
top:Face	Wall, Roof, Floor	IfcWall, IfcSlab	BuildingSurface
top:Aperture	Door, Window	IfcDoor, IfcWindow	FenestrationSurface

Figure 98. Cypher query performed on Neo4j to retrieve the spaces contained in the zones of the building.

```
MATCH (building:bot_Building {UID: 'building02'})
MATCH (building)<-[:brick_hasLocation]-(zone:brick_Energy Zone)
MATCH (zone)<-[:brick_hasLocation]-(space:bot_Space)
RETURN building.UID as BUILDING,
       zone.UID as ZONE,
       space as SPACE
```

Figure 99. Cypher query performed on Neo4j to retrieve energy-related KPIs of zones.

```
MATCH (building:bot_Building {UID: 'building02'})
MATCH (building)<-[:brick_hasLocation*.. ]-(zone:brick_Energy_Zone)
MATCH (zone)<-[:eko_hasAssociatedObject]-(zoneKPI:btwin_KPISet)
MATCH (zoneKPI)-[:kpi_relatedScenario]->(scenario:kpi_Scenario)
WHERE zoneKPI.UID CONTAINS "energy"
RETURN DISTINCT
       building.UID AS BUILDING,
       zone.UID AS ZONE,
       zoneKPI.UID AS KPI,
       zoneKPI AS KPI_DATA,
       scenario.UID AS SCENARIO
```

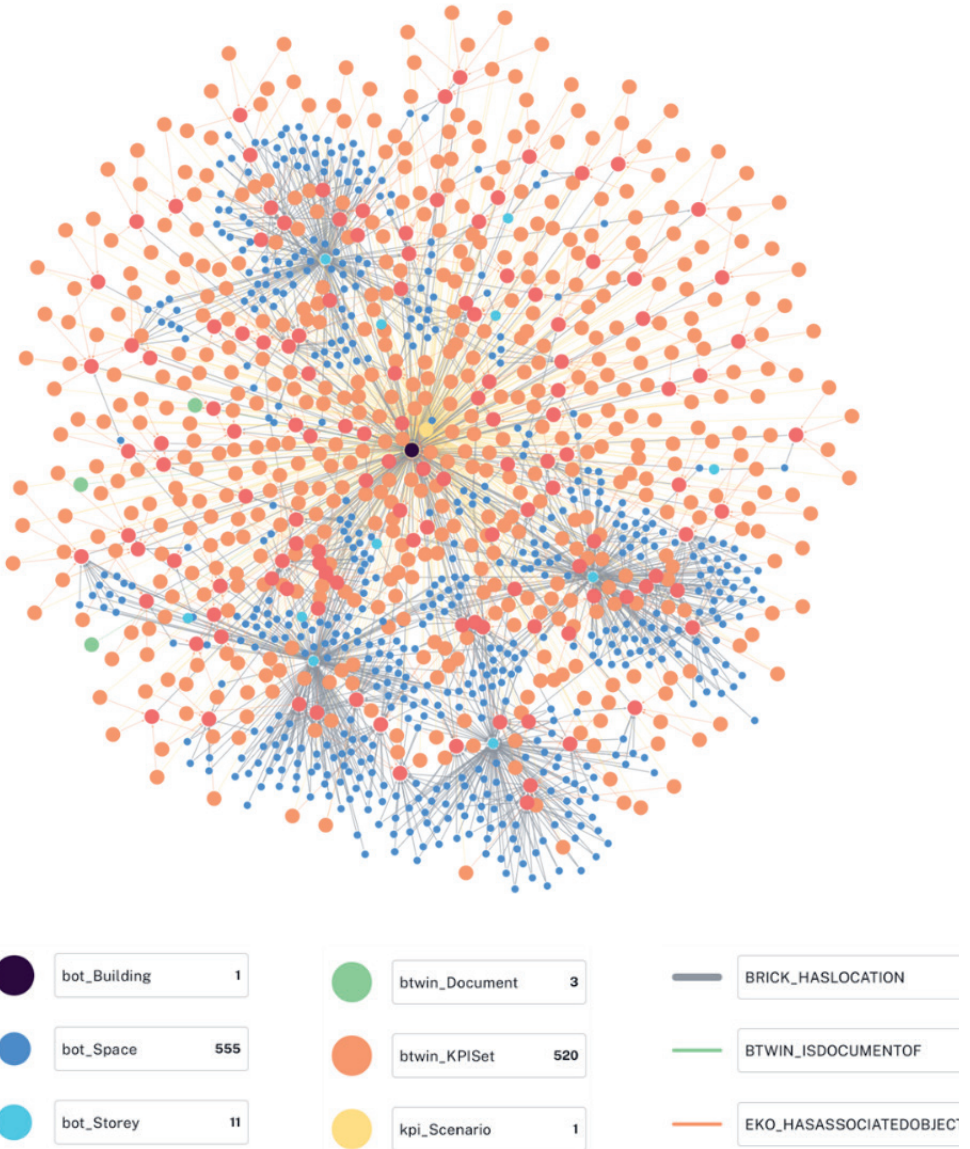


Figure 100. Neo4j integrating the spatial and performance hierarchy for the whole building. Nodes and edges are coloured according to their semantic labels.

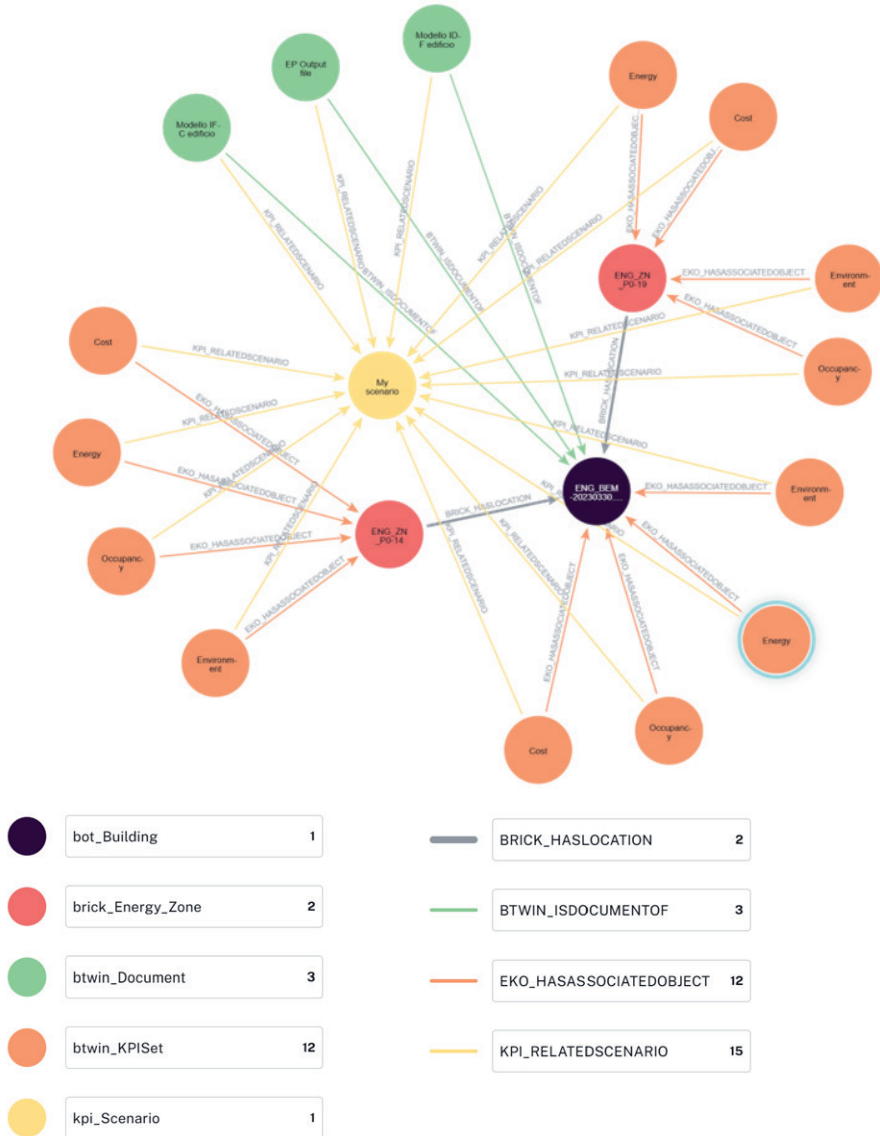


Figure 101. Portion of the KG created for the demonstration explaining the relationship between the spatial and performance hierarchy.

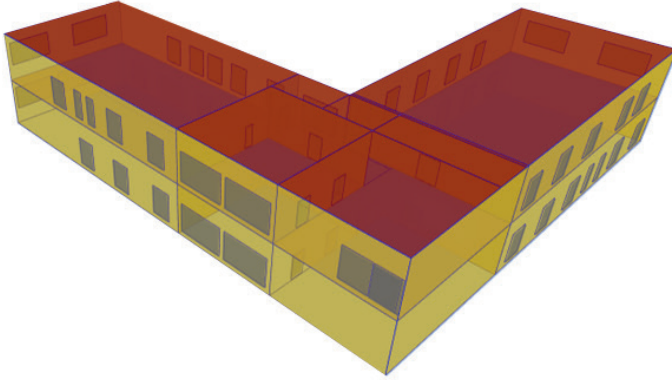
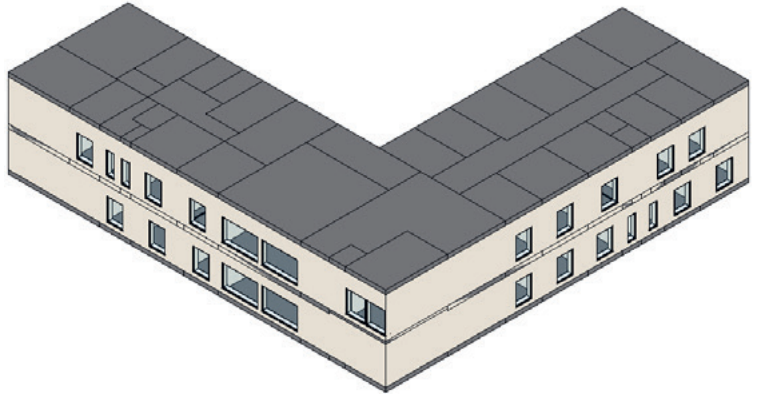


Figure 102. EnergyPlus IDF model of a part of the case study building and corresponding spatial hierarchy in the Neo4j graph.



Figure 103. IFC model of a part of the case study building and corresponding spatial hierarchy in the Neo4j graph.



8.4 Results and outcomes

Building upon this data storage system, a set of interactive dashboards was developed to visualise these KPIs directly on the building's 3D geometry. This interface allows users to explore energy performance across different levels of detail, by storey, zone, or space. Within the dashboards, users can query the KG intuitively, for example, to identify spaces with the highest energy demand per square metre or to compare energy costs among different zones (Figure 104).

The analysis focused in particular on classrooms, which were considered the most representative spaces within the building for investigating relationships between energy and occupancy. To enable comparison across different spaces, a *k*-means clustering algorithm was applied to group classrooms by floor area and peak occupancy (Figure 105)¹⁴. This clustering facilitated the visualisation of KPIs by cluster, as shown in the figures below.

The simulation results were visualised on a zone-by-zone and day-by-day basis to enable further analysis. For example, a detailed examination of a typical winter day revealed a strong correlation between heating demand and occupancy levels (Figure 106).

In some classrooms, occupancy peaks coincided with almost negligible heating demand, suggesting that occupancy-driven control strategies could help maintain satisfactory indoor thermal conditions while avoiding unnecessary heating, thereby reducing operational costs. Conversely, during the summer period, peaks in external temperature coincided with maximum occupancy levels. Under these conditions, allocating classrooms with lower solar gains for activities occurring during peak hours becomes essential to mitigate thermal discomfort and reduce cooling energy consumption.

Cluster-level analysis also helped identify outliers and rank spaces according to economic or environmental KPIs (Figure 107). For instance, some medium-sized classrooms displayed disproportionately high heating demand, suggesting possible inefficiencies in occupancy scheduling.

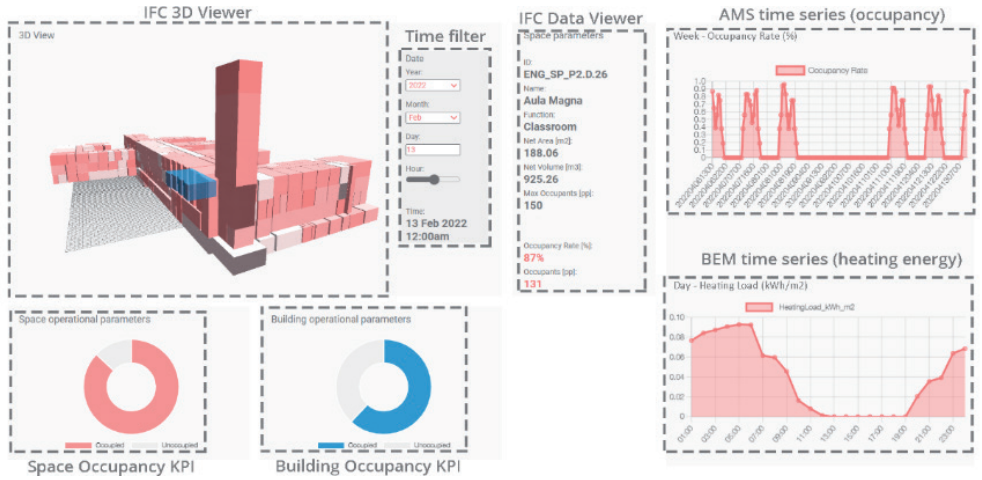


Figure 104. Prototype dashboard to access and visualise data about the School of Engineering of Bologna.

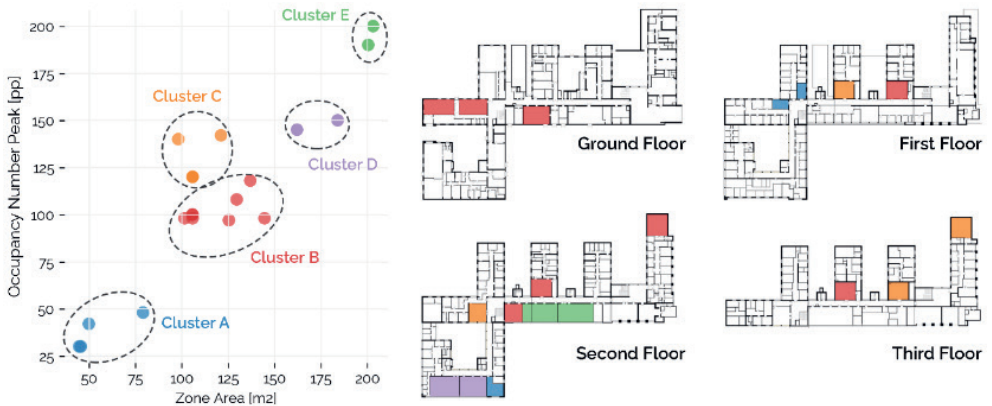


Figure 105. K-means clustering of the classrooms by area and peak occupancy number.

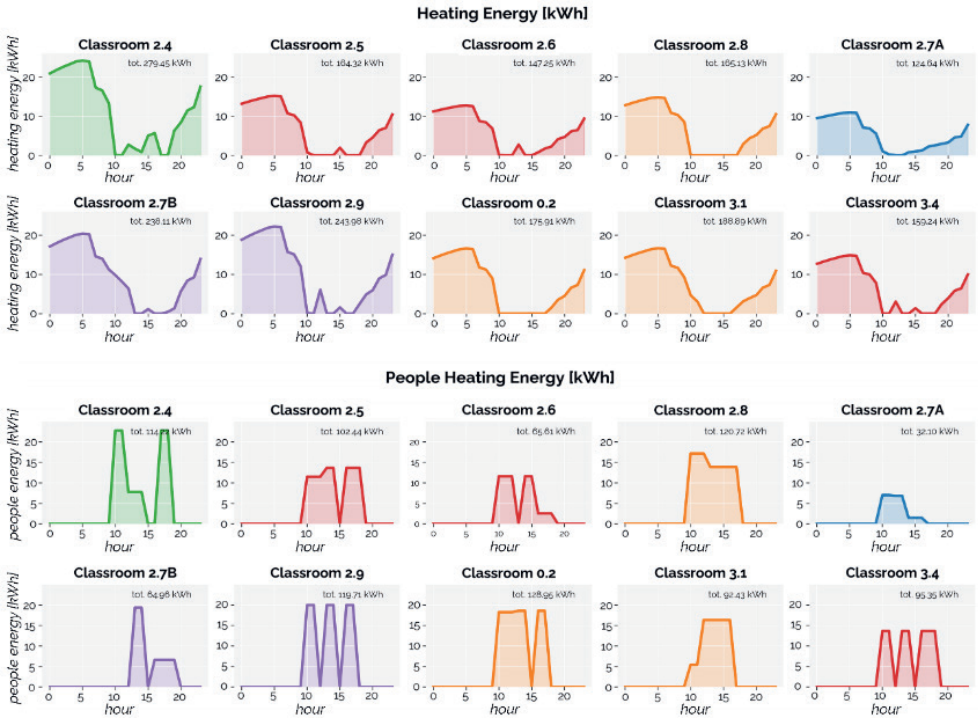


Figure 106. Visualisation of hourly heating demand for some classrooms for the specified date and time. Data are coloured according to the clusters identified in Figure 105.

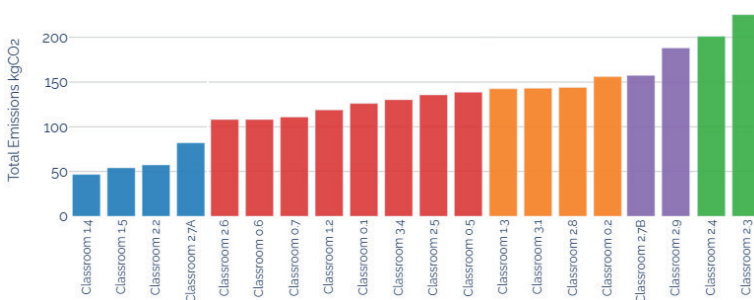


Figure 107. Total emissions for each classroom during the analysis day. Data are coloured according to the clusters identified in Figure 105.

Overall, this use case extended the methodological framework presented in the previous chapters by demonstrating how KGs can facilitate interoperability across different modelling environments and how hybrid data architectures can serve as the knowledge bases of modern decision support systems by managing both static and dynamic data.

In the application, KGs were not merely repositories of geometric data or tools for spatial analysis; instead, they functioned as comprehensive data hubs linking schedules, simulation outputs, and performance indicators. This ability to represent relationships across multiple domains (geometry, topology, construction, and performance) at different scales is essential for performance-based building management. The advantage of using simulated data was the possibility of obtaining fine-grained analyses, certainly less precise than those based on sensor data, but more extensive across the whole building. In this context, each stakeholder, depending on their specific role and responsibilities, can access data at different levels of granularity to support informed decision-making.

In the following chapter, the methodology introduced here will be further developed, presenting a workflow that employs graphs for both spatial analysis (safety graphs) and interoperability (between BIM, BEM, and BSM). Moreover, this next stage extends the use of graphs beyond spatial dimensions to include temporal scales, introducing the concept of 'temporal graph' for scenario modelling in support of what-if decision-making.

Notes

¹ Building performance simulation is the field concerned with reproducing and forecasting various aspects of a building's behaviour through computer-based mathematical models grounded in physical principles and engineering methods. It is a rapidly evolving discipline, characterised by active research, continuous development, and growing adoption in professional practice. (de Wilde 2023)

² As outlined in the previous chapter, TBIM offers a modelling approach that differs from traditional, product-focused BIM methods. Instead of depicting buildings through detailed construction elements, it represents them as assemblies of spatial units. This transition toward a space-oriented, topological model results in representations that are lightweight in geometry but rich in semantic content, making them particularly suitable for integration into Building Performance Simulation (BPS) processes. For further details, see the article titled 'Topological BIM for Building Performance Management.' (Masafra et al. 2024)

³ The energy usage and intensity of buildings on a higher education campus are influenced by several factors, including the climate, building systems, construction type, but also occupancy conditions (Wadud et al. 2019). Occupancy variables, such as the presence of students and staff members and the activities they do, can significantly impact energy consumption. For example, Gui et al (Gui et al. 2021), in 'Reducing university energy use beyond energy retrofiting: The academic calendar impacts', showed how the academic calendar, which regulates the occupancy condition of campus buildings, determines their energy consumption in Australian universities. In contrast, Mosteiro-Romero et al. (Mosteiro-Romero et al. 2023) demonstrated how demand-driven building system controls can save energy in university offices and how such strategies are important to adapt building use to climate change in summer conditions in Singapore.

⁴ In terms of size, the Alma Mater manages roughly one million square metres of floor area, making it the largest public property holder in the Emilia-Romagna region, and accommodates over 80,000 people, a population comparable to that of a mid-sized Italian provincial city.

⁵ Constructed between 1932 and 1935, it is among the earliest 20th-century buildings to be officially listed in the city (Gulli and Predari 2018; Predari et al. 2021). Recognised as a notable example of local rationalist heritage, it stands out for its use of industrial systems and materials, innovative finishes, and absence of ornamentation. Covering 19,200 m² of net floor area across four levels, the building can host up to 5,000 occupants with approximately 2,500 students using the facility five days a week for eleven months each year.

⁶ During that period, a noticeable decrease in building occupancy was observed, resulting in many spaces remaining unoccupied for several months, as demonstrated by the evident reduction of domestic hot water consumption. However, there was no corresponding decrease in natural gas consumption for heating, which remained similar to pre-COVID years, indicating a critical mismatch between energy demand and space usage.

⁷ The number of students attending each course was determined using University Planner, a publicly available service that provides university schedules, including the time, classroom, and course for each lesson. The expected

occupancy for each class was estimated by counting enrolled students in the University's Virtual Learning Environment. For other functional areas (e.g., research and administration offices), occupancy profiles were defined by adapting ASHRAE schedules to the university's working hours. In total, 38 distinct schedules were considered, covering classrooms, offices, food services, circulation spaces, support spaces, housing, and laboratories.

⁸ For the analysis, it was assumed that each zone was equipped with thermostats. The heating setpoints were set to 20 °C from 7:00 a.m. to 8:00 p.m., and 16 °C during the night. Lighting was assumed to remain on continuously from 8:00 a.m. to 8:00 p.m., while occupancy was set to the planned number of people based on the university's applications. Mechanical ventilation was not considered because the building relies on natural ventilation. Cooling was included only for offices, since most classrooms and other spaces are not equipped with cooling systems. Office cooling was assumed to be active during summer occupancy periods, with a setpoint temperature of 26 °C. A similar approach was adopted for lighting, using a setpoint of 300 lux, and for equipment usage, both of which were aligned with occupancy schedules. The natural ventilation rate was assumed to be 0.3 air changes per hour, based on Semprini et al. (Semprini et al. 2016), derived from an energy audit of the same building conducted in 2016. Natural ventilation schedules were also synchronised with occupancy schedules.

⁹ Energy needs were calculated by aggregating and summing the observations for lighting electricity, total cooling energy, equipment electricity for the electrical component, and total heating energy for the gas component. Likewise, the KPIs related to overall energy demand were derived by aggregating these observations.

¹⁰ For cost estimation, the energy data were multiplied by the unit prices for gas and electricity from the building's utility bills for 2022–2024, yielding 0.054 EUR/kWh for gas and 0.159 EUR/kWh for electricity. The conversion from standard cubic metres (Smc) reported in the bills to kilowatt-hours (kWh) for natural gas was performed using a conversion factor of 10.69, which accounts for the gas's calorific value.

¹¹ Greenhouse gas emissions were estimated using conversion factors of 0.49 kg CO₂eq/kWh for electricity and 0.25 kg CO₂eq/kWh for gas, in accordance with the emission coefficients provided by the International Energy Agency (IEA).

¹² The occupancy rate was calculated as the average occupancy for each zone, expressed as a percentage (0 = unoccupied, 1 = fully occupied), based on the occupancy schedules defined in the energy model.

¹³ The containment relationship between spaces (from the BIM) and thermal zones (from the BEM) was automatically established through a connector developed using Topologicpy APIs. This connector uses a set of custom functions to determine which spaces belong to which zones – a process made possible by the fact that both models share a common coordinate reference system.

¹⁴ Cluster A comprises classrooms with an area of less than 80 sqm and accommodating up to 50 people, representing the smaller-sized classrooms in the building. Cluster E consists of classrooms with an area larger than 190 sqm and accommodating up to 175 occupants, representing larger classrooms. Cluster B includes classrooms with an occupancy range of 80 to 120 people

and an area between 100 and 150 sqm. Cluster C contains classrooms with an occupancy range of 125-150 people and an area of 100-120 sqm. Lastly, Cluster D represents classrooms with an occupancy range of 140 to 150 people and an area between 160 and 180 square meters.

9. SCENARIO GRAPHS FOR MULTICRITERIA DECISION MAKING

Abstract

Shifting toward digital, performance-oriented management requires not only seamless data integration but also the development of systems that actively support strategic decision-making.

One technique well suited to this purpose is *scenario analysis*, which consists of modelling and simulating hypothetical intervention scenarios to identify those with the most beneficial impacts on management. This approach, particularly valuable in complex systems where decisions produce interrelated consequences, introduces a temporal dimension into data governance that requires organising and processing information not only about past conditions, but also about the current and potential future states.

As the final demonstrative chapter of the book, this contribution presents a scenario-modelling workflow that integrates BIM (Building Information Modelling), BEM (Building Energy Modelling), and BSM (Building Safety Modelling) data within a Knowledge Graph (KG) framework to evaluate the impacts of occupancy management on both energy demand and occupant safety across different scenarios.

The work extends the case study introduced in Chapter 6 and builds upon research conducted within the DIGITMAN project (PRIN 2022 programme, ID 2022JMRX2A), whose components are here thanked: Marco D’Orazio, Gabriele Berardini, and Guido Romano (Università Politecnica delle Marche), Graziano Salvalai, Manuela Grecchi, and Roberto Villa (Politecnico di Milano), Giorgia Predari and Riccardo Gulli (Università di Bologna).

9.1 Contextualisation and use case definition

This chapter concludes the demonstrative part of the book by presenting the case study with the highest level of data-management complexity. The two preceding chapters applied the methodological foundations of *Buildings as Networks* to different contexts – respectively, integrating BIM with sensor data (Chapter 6) and coupling BIM with BPS data (Chapter 7) – to monitor and assess the performance characteristics of selected buildings. Here, the focus shifts from retrospective analyses of historical data to forward-looking temporal graphs that support multi-criteria decision-making; in other words, from monitoring to prediction.

The objective of the work is threefold. First, the chapter demonstrates how a KG approach can combine static BIM data with both energy and safety simulations to evaluate the impacts of occupancy management actions. Second, it illustrates how to manage data under diverse hypothetical temporal conditions within a predictive framework, extending the temporal dimension from past time-series data to projected what-if scenarios. Third, it shows how two distinct uses of graphs can coexist within the same workflow: the topological/spatial function (introduced in Chapters 5 and 6) and the data-integration function (presented in Chapters 7 and 8).

Furthermore, unlike previous case studies that focused on single domains, this chapter demonstrates the scalability of the proposed methodology across multiple disciplines. In particular, it introduces an integrated workflow for analysing three key aspects of building management, i.e. occupancy, energy, and safety.

In this study, all objectives converge on the development of a decision-support system (DSS) conceived within the DIGIT-MAN project, designed to enable what-if simulations based on a multicriteria logic. The DSS, equipped with a user-friendly dashboard, allows users to select buildings and alternative occupancy scenarios and to visualise their respective impacts on both energy performance and safety. The system is specifically tailored to support long-term what-if analyses, enabling users to compare hypothetical interventions – such as timetable adjustments or the re-functionalisation of spaces – and to under-

stand their managerial implications.

Graphs are employed in this research to address two key challenges. The first concerns the need to correlate spatial and construction elements with performance metrics across multiple analytical scenarios, accommodating different levels of granularity – from the overall building scale to energy zones, fire compartments, individual spaces, and even single door components. The second challenge lies in overcoming interoperability barriers between BIM, BEM, and BSM models, thereby enabling the federation of multiple specialist models within a single, coherent knowledge system. A direct implication of this approach is the ability to manage not only a single operational condition but also multiple hypothetical ones, each representing a distinct management scenario.

The key contribution of this work is an open-source workflow that first converts BIM models into interoperable energy and safety models, then links them into a single knowledge base via a series of graph transformations.

9.2 Case study: occupancy management in higher education buildings

The demonstrative application focuses on the Lecco campus of the Politecnico di Milano, already introduced in Chapter 6. Specifically, it examines two multi-storey buildings that are physically connected at the first-floor level – meaning that the energy performance and safety conditions of one directly influence those of the other. The buildings accommodate classrooms, study areas, and supporting facilities and are occupied daily by students, researchers, and academic staff in accordance with the university timetable.

Their continuous use and high occupancy variability make these buildings particularly suitable for exploring the interrelationships among spatial configuration, energy demand (as discussed in Chapter 8), evacuation safety, and occupancy-based performance assessment.

9.3 Methods and tools

9.3.1 Workflow articulation

The methodology adopted to pursue the research objectives is organised into three main phases, described in the following sections:

1. Data acquisition;
2. Data modelling;
3. Data processing;
4. Data integration;
5. Data visualisation and microservice development.

9.3.2 Data acquisition

The data acquisition phase, conducted by the research team at Politecnico di Milano as part of the DIGITMAN project, relied on the retrieval and examination of documentation provided by the property managers. In addition to the information collected about the case study for modelling fire-safety aspects, already discussed in Chapter 6, further data were obtained to support the energy-modelling process, including:

- Geometric, material, and performance characteristics of the envelope systems, acquired through technical data sheets describing the installed construction components;
- Technical documentation and related information on the mechanical systems, mainly consisting of air-handling units responsible for heating, cooling, and ventilation;
- Historical data about energy consumption for HVAC, provided to supply detailed information on actual energy consumption for the main environments;
- Temperature and humidity data, acquired by sensors in some representative rooms, employed to calibrate the energy model.

9.3.3 Data modelling

All collected data were then used in the data-modelling phase, which involved developing three main models:

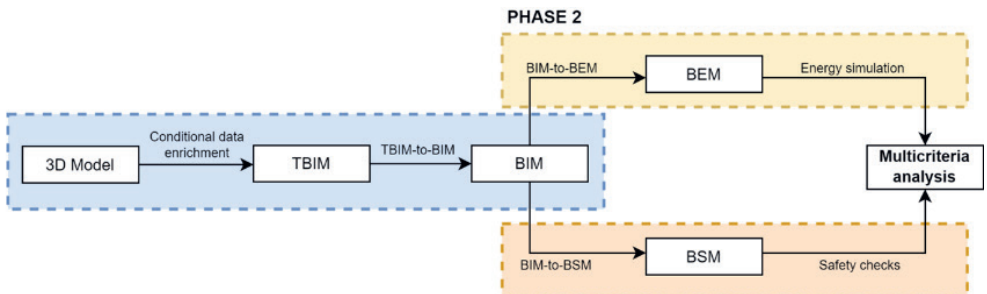
- The *Topological BIM* (TBIM), provided in IFC format, defined the hierarchical and topological structure of the building.
- The *Building Safety Model* (BSM), in the Topologic JSON format, enables the performance of safety checks, as described in Chapter 6.
- The *Building Energy Model* (BEM), in EnergyPlus IDF format, supports energy simulations aimed at assessing the energy requirements of the spaces and their relationship to occupancy metrics.

The processes used to generate the TBIM and BSM for this case study have already been detailed in Chapter 6, specifically in Section 6.3.3. In addition, the following paragraphs describe the BIM-to-BEM conversion process applied in this context, as well as the subsequent modification of the BSM and BEM files for scenario modelling. Figure 108 provides a high-level overview of the workflow structure.

Topological BIM

The first step of the data-modelling phase involved developing, for each building, the Topological BIM (TBIM) model, whose process was already introduced in Section 6.3.3. Into this model, all the information required for fire-safety modelling was incorporated. Likewise, all data necessary for the energy-modelling activities were integrated according to the datasets reported in Chapter 6.

Figure 108.
Methodological
overview.



Building Safety Modelling

The BSM generation process consisted of a BIM-to-BSM workflow that converts the IFC model into three Topologic graphs representing the building's evacuation system. When combined, these graphs form the BSM, a comprehensive analytical framework for conducting safety assessments. This process was previously detailed in Section 6.3.3.

Building Energy Modelling

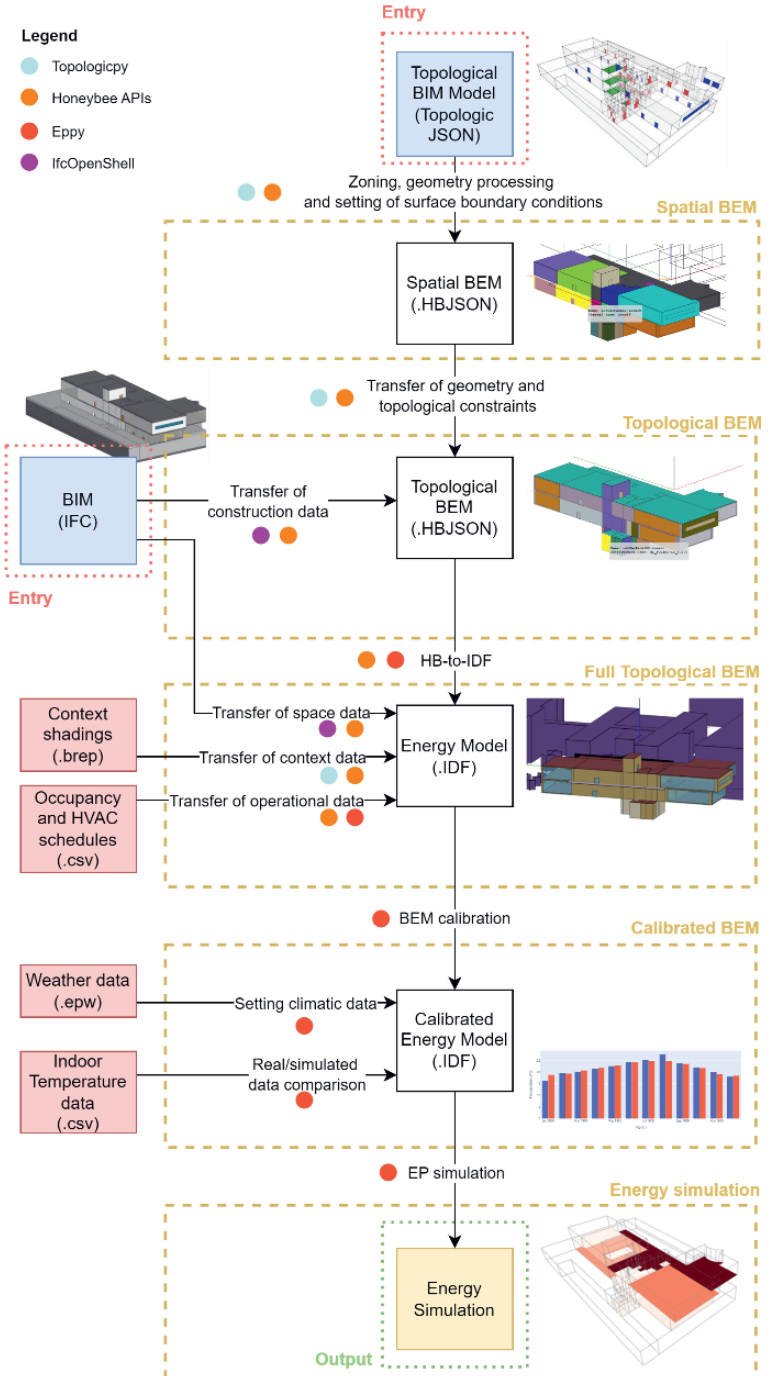
The BIM-to-BEM conversion was performed using a Python-based algorithm that aligns the element classes and properties defined in Topologic and IFC with those of Ladybug Tools - through the Honeybee (HB) APIs (Ladybug Tools 2025) - and EnergyPlus (EP) - via the Eppy library (Philip 2020). The BIM-to-BEM process, depicted in Figure 109, is explained below.

The main steps followed in the BIM-to-BEM process were:

- Zoning and geometric definition
- Setting boundary conditions
- Characterization of constructions and openings
- Context and shading modelling
- Definition of simulation parameters
- Infiltration modelling
- Schedule definition
- Integration of operational data
- Weather data input

Zoning and geometry. The first step in the BIM-to-BEM conversion process involved zoning and geometry processing. To achieve this, the `Cells` of the TBIM (in the Topologic JSON format) were aggregated into `Cluster` objects, each representing a thermal zone within the BEM¹. Once the zones were defined, aggregation functions were applied to transfer energy-related parameters from the individual cells to the corresponding thermal zones. Subsequently, by integrating TopologicPy with the HB APIs, the `Topologic Face` objects were converted into `HB Faces`, and the `Topologic Aperture` objects into `HB Apertures`. For each thermal zone, these HB entities were then passed to the `HB Room` method, generating the energy zones that together constitute the base `HB Model`. Results of the zoning process are depicted in Figure 110.

Figure 109. BIM-to-BEM workflow.



Boundary conditions. After generating the HB Model, the boundary conditions of the HB Faces that constitute the model (i.e. *outdoor*, *adiabatic*, and *ground*) were automatically computed (Figure 111). Subsequently, the faces of the building that were adjacent to other buildings are manually set to *adiabatic*.

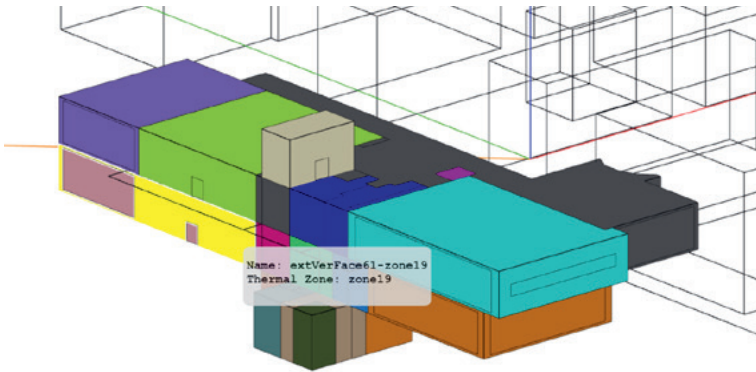


Figure 110. BEM model of Building 10. Faces coloured by zone.

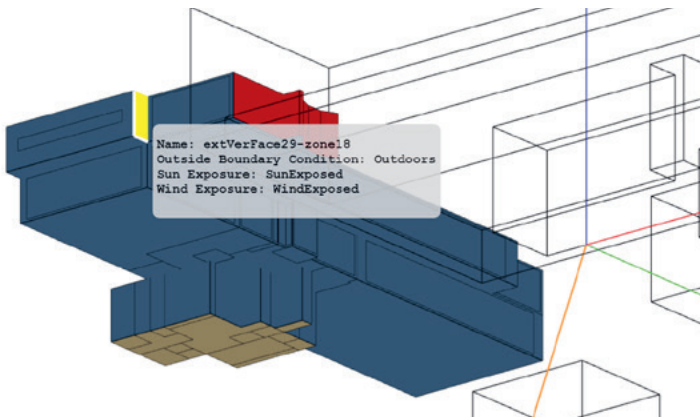


Figure 111. BEM model of Building 10. Faces coloured by boundary condition (red: adiabatic, blue: outdoors, yellow: ground).

Construction characteristics and openings. Construction data were first added to the base HB Model (Figure 112). The properties of the opaque envelope were extracted directly from the IfcMaterialLayerSets associated with IfcWall and IfcSlab entities in the IFC model, retrieving the thickness, conductivity, density, and specific heat of the IfcMaterials composing each layer set. These IfcMaterials were then converted into HB EnergyMaterial objects, subsequently combined into HB OpaqueConstruction assemblies. Next, the thermal characteristics of the building apertures were transferred to the corresponding HB Model's components. Glazing-related data – such as thermal transmittance, solar heat gain coefficient, and visible transmittance – were extracted from the IfcWindowType and IfcDoorType entities in the IFC model and converted into HB WindowConstruction and HB Door objects, respectively.

Context shadings. To simulate shade effects on the model, context elements that cast shadows on the building (e.g., vegetation, surrounding buildings, shading systems) were modelled within a 3D modelling environment, exported as BRep entities, converted into Topologic Faces, and then linked to the BEM as HB Shade elements (Figure 113).

Simulation parameters. The simulation parameters were then configured through Eppy. These include the simulation run period, the shadow calculation method, the daylight saving calculation method, and the output variables, which refer to the KPIs to be analysed through the simulation (i.e., zone heating and cooling energy, zone electric energy for lighting and equipment, and zone people occupant count). The HB Model, enriched with this information, was exported as an IDF file. From this point onwards, additional information was assigned to the IDF using the Eppy library.

Infiltration. The infiltration rate of the building was set in the IDF by assigning a ZoneInfiltration:DesignFlowRate object to each building zone in Eppy².

Schedules. Input schedule data was sourced from JSON files containing schedules, formatted as required by the HB APIs for

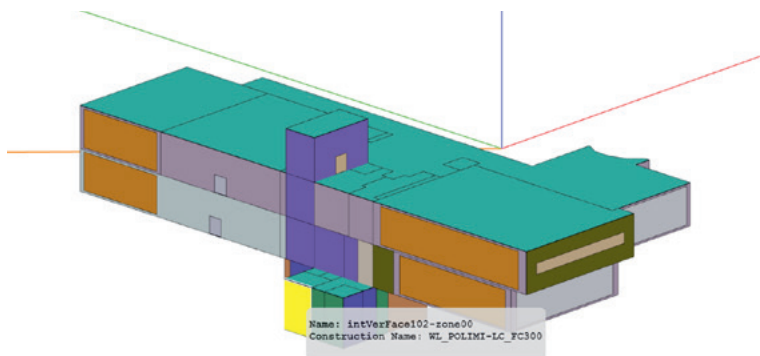
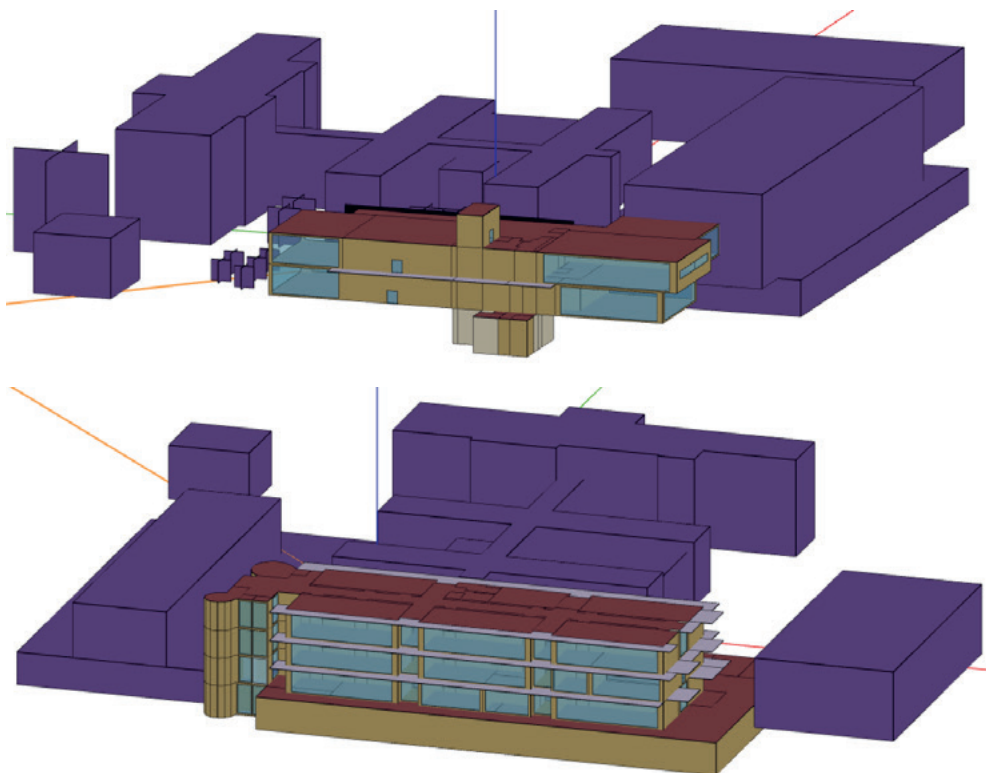


Figure 112. BEM model of Building 09. Faces coloured by construction type.

Figure 113: IDF models of the selected pilot site and context shadings. Building 10 on the top, Building 09 on the bottom.



modelling ScheduleRulesets. A Python function was then created to transform this data into an IDF Schedule:Compact objects. The schedules here considered include fraction types, such as occupancy schedules (occupancy density in each zone for every hour of the year) and lighting and equipment schedules (percentage of power usage for lighting and equipment per hour) and boolean types, such as the availability schedules for heating, cooling, and ventilation systems³.

Operational data. Static operational data were instead transferred from the IFC to the IDF. Occupancy data, assigned zone by zone to IDF People elements, included people per floor area in operational conditions and the corresponding schedules. Lighting data, assigned instead to IDF Lights objects, consists of the electric power density of lighting devices and relative schedule names. Similarly, equipment data, assigned to IDF ElectricEquipment objects, includes the electric power density used by electrical appliances and relative schedules. Instead, thermal and humidity control data are assigned to IDF HVACTemplate:ZoneIdealLoadsAirSystem objects. These include temperature and humidity setpoints, related schedules, and HVAC system availability schedules. Finally, ventilation data are assigned to DesignSpecification:OutdoorAir objects, which specify the method for calculating outdoor air changes and the outdoor air-flow per person and per zone floor area.

Weather data. Climatic data were input into the BEM using the EnergyPlus Weather (EPW) format. In this case, air temperature, relative humidity, atmospheric pressure, wind speed, wind direction, and global solar radiation data collected from a weather station close to the analysed building over the last three years were used (Figure 114).

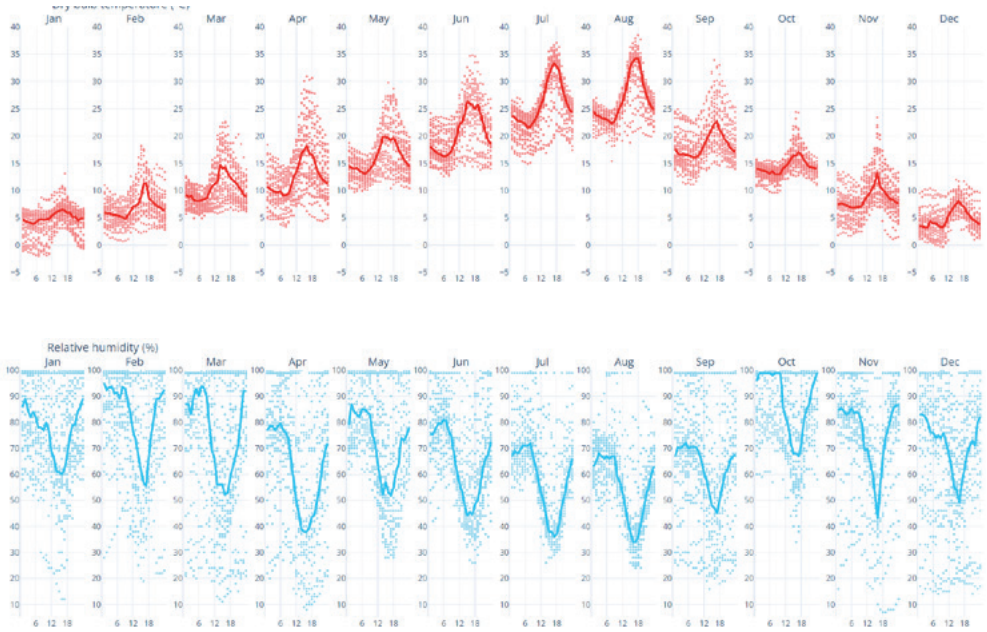


Figure 114: Daily chart with temperature and humidity values. EPW analysis in CBE Climate Tool. (Betti et al. 2024)

9.3.4 Data processing

After generating the baseline models, they were processed to obtain valuable performance data. The data-processing phase comprised several sub-steps: BEM calibration, needed to ensure that the BEM accurately reproduced the measured operational behaviour; scenario modelling, in which alternative configurations and boundary conditions were defined; and execution of the simulations, through which the resulting performance indicators were computed.

BEM calibration

The BEM calibration was carried out to ensure that the model accurately represents the building's thermal behaviour by minimising discrepancies between predicted and measured data.

Because detailed energy consumption data were not available at the building or zone level – but only at the campus level – the calibration was conducted by comparing sensor temperature readings with simulated indoor temperatures in a couple of classrooms equipped with a sensor network by the POLIMI group⁴. The calibration involved iteratively modifying temper-

ature setpoints and operational schedules for both the heating and cooling seasons to achieve the closest match between simulated and measured indoor temperatures.

The comparison between real and simulated data was conducted using several statistical indicators, both standard (Mean Squared Error, MSE, and Root Mean Squared Error, RMSE⁵) and customised (Mean Temperature Difference in the Week, MTDW). Four representative weeks were analysed – winter, summer, spring, and autumn – selected from the EPW climatic dataset.

Model debugging and parameter adjustments were performed iteratively. Technical corrections included revising ventilation schedules, redefining window shading systems, optimising solar distribution methods, refining simulation timesteps, and converting temperature setpoints into `Schedule:Compact` objects for more accurate control. Heating and cooling setpoints were then fine-tuned (22°C with setbacks at 18°C for heating and 24°C for cooling) and linked to daily operation schedules.

After calibration, the discrepancies between real and simulated data decreased substantially. RMSE values reduced from approximately 3–5°C before calibration to around 1–2°C after calibration across all buildings and seasons. Similarly, average monthly temperature differences fell within $\pm 1.0^\circ\text{C}$, confirming that the calibrated BEM reliably reproduced the thermal performance of the analysed buildings and could be used for subsequent scenario-based energy simulations (Figure 115, Figure 116, and Figure 117).

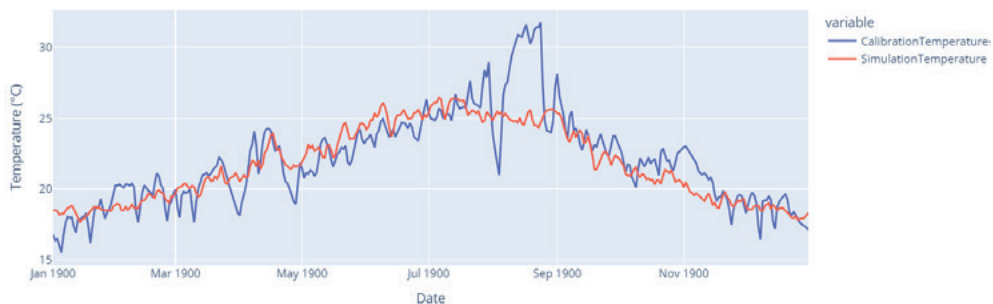


Figure 115: Daily Real vs Simulated data after the model calibration, Building 10.

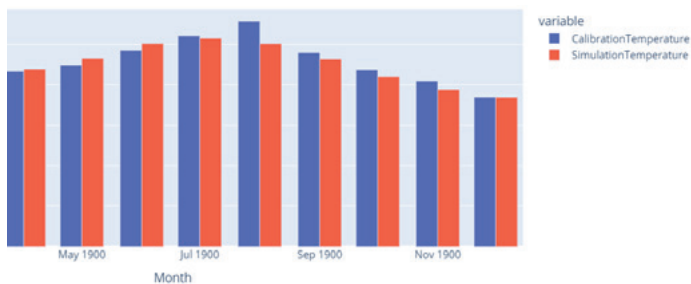


Figure 116: Monthly Real vs Simulated data after the model calibration, Building 10.

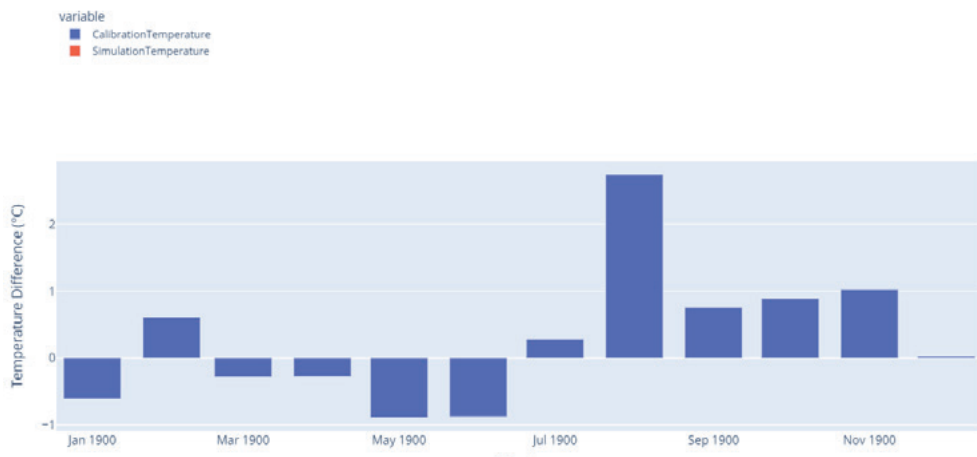


Figure 117: Monthly difference between mean real temperature and mean simulated temperature, after the model calibration, Building 10.

Scenario modelling

Based on the BSM and the calibrated BEM, scenarios were modelled to represent hypothetical changes in occupancy management and simulate their impacts⁶.

In the study, the selected what-if scenarios to be simulated explore variations in the occupancy characteristics of the main building spaces. The intention is to investigate questions such as:

- *What happens if classrooms are converted into offices?*
- *What happens if the capacity of specific spaces is increased or reduced?*
- *What happens if a demand-driven ventilation system is introduced?*
- *What happens if the academic timetable or space-use schedules are modified?*⁷

To represent such effects, four categories of modifications to the building's current configuration were defined for modelling the scenarios. These categories correspond to plausible management actions that facility managers may implement during the operational phase of the asset:

1. *Variation in space capacity*: altering the allowable number of occupants within specific spaces, for instance, in response to health and safety regulations, which directly affects ventilation rates as well as heating and cooling loads.
2. *Adjustment of occupancy schedules*: modifying the temporal patterns of space utilisation (e.g. daily or weekly schedules), thereby influencing system operating hours and peak energy demand periods.
3. *Space repurposing*: changing the functional use of rooms (for example, converting classrooms into offices or laboratories), which alters occupancy profiles, internal gains, and environmental control requirements.
4. *Implementation of demand-driven ventilation*: replacing constant air volume systems, typically designed for maximum occupancy, with variable air volume systems that dynamically adjust airflow according to actual occupancy or indoor environmental conditions.

These categories may be combined within the same what-if configuration, allowing the development of complex, multi-variable scenarios for comprehensive energy and safety perfor-

mance assessment. Table 20, Table 21, and Table 22 provide an exemplification of this approach.

Question statement	Question Subject	Question Action	Scenario description
What If	the academic calendar or schedule structure	is changed?	Altering the academic calendar affects space-use patterns and can alter energy demand and operational scheduling.
What If	working-from-home (WfH)	is introduced in offices?	Large-scale adoption of WfH can reduce space demand, enable flexible strategies, lower operational costs, and produce environmental benefits.
What If	blended learning or partial distance learning	are introduced?	Similar to WfH, increased use of blended or remote learning reduces the need for physical space, supports flexible space planning, and contributes to environmental goals by decreasing travel-related emissions.
What If	spaces	cannot be used for a long time?	Prolonged non-use of spaces can lead to energy waste if systems lack adaptive controls. Fixed schedules or inflexible systems may continue to operate even when occupancy is low, thereby increasing inefficiency.
What If	the maximum capacity of spaces	is changed (i.e. increased)?	Space capacity may be altered due to health guidelines or safety regulations. Such changes affect scheduling, occupancy rates, and operational use of spaces.
What-if	the intended use	is changed (i.e. riskier)?	Similarly, changing the function of a space may conflict with adjacent uses or building systems.

Table 20. Example what-if scenarios in the context of occupant-based higher education building management. Variations in occupancy – Energy perspective.

Table 21. Example what-if scenarios in the context of occupant-based higher education building management. Occupant-centric controls – Energy perspective.

Question statement	Question Subject	Question Action	Scenario description
What If	the academic calendar or schedule structure	is changed?	Altering the academic calendar affects space-use patterns and can alter energy demand and operational scheduling.
What If	working-from-home (WfH)	is introduced in offices?	Large-scale adoption of WfH can reduce space demand, enable flexible strategies, lower operational costs, and produce environmental benefits.
What If	blended learning or partial distance learning	are introduced?	Similar to WfH, increased use of blended or remote learning reduces the need for physical space, supports flexible space planning, and contributes to environmental goals by decreasing travel-related emissions.
What If	spaces	cannot be used for a long time?	Prolonged non-use of spaces can lead to energy waste if systems lack adaptive controls. Fixed schedules or inflexible systems may continue to operate even when occupancy is low, thereby increasing inefficiency.
What If	the maximum capacity of spaces	is changed (i.e. increased)?	Space capacity may be altered due to health guidelines or safety regulations. Such changes affect scheduling, occupancy rates, and operational use of spaces.
What-if	the intended use	is changed (i.e. riskier)?	Similarly, changing the function of a space may conflict with adjacent uses or building systems.

Question statement	Question Subject	Question Action	Scenario description
What If	the maximum capacity of spaces	is changed (i.e. increased)?	Modifying the maximum room capacity be considered for future reuse strategies. However, such changes might not be feasible due to spatial constraints, emergency layout limitations, or incompatibility with adjacent space overall building configuration. Computer-planned and currently allowed capacities help identify potential design conflicts, trigger alerts that may require broader renovation planning. This assessment conducted independently or alongside evacuation-related analyses. A top-down approach may help resolve inconsistencies in spatial allocation and intended use.
What-if	the intended use	is changed (i.e. riskier)?	Changing the function of a space can alter occupancy conditions and increase number of occupants, potentially in higher safety risks. These changes should be carefully evaluated for their impact on building's overall safety.

Table 22. Example what-if scenarios in the context of occupant-based higher education building management. Variations in occupancy – Safety perspective.

In particular, the following what-if scenarios were defined for the selected case study:

- *Scenario 0 – Current State (2024)*: baseline scenario with lecture halls in 2024 with Constant Air Volume (CAV) systems controlled by volume flow.
- *Scenario 1 – Classrooms CAV-Person*: same as Scenario 0, but CAV systems are controlled based on airflow per person.
- *Scenario 2 – Classrooms VAV-Person*: same as Scenario 0, but with Variable Air Volume (VAV) systems that modulate flow based on occupancy (maintaining minimum flow when unoccupied as per regulations).
- *Scenario 3 – Classrooms +20% Capacity*: classroom seating capacity is increased by 20%.

- *Scenario 4 – Classrooms -20% Capacity*: classroom seating capacity is reduced by 20%.
- *Scenario 5 – Classrooms Fully Occupied*: classrooms are assumed to be 100% occupied during all operating hours.
- *Scenario 6 – Classrooms Half Occupied*: classrooms are assumed to be 50% occupied during all operating hours.
- *Scenario 7 – Classrooms 75% Occupied*: classrooms are assumed to be 75% occupied during all operating hours.
- *Scenario 8 – Classrooms 25% Occupied*: classrooms are assumed to be 25% occupied during all operating hours.
- *Scenario 9 – Closed Offices CAV-Volume*: closed office spaces with CAV systems controlled by volume flow.
- *Scenario 10 – Closed Offices CAV-Person*: closed office spaces with CAV systems controlled per person.
- *Scenario 11 – Closed Offices VAV-Person*: closed office spaces with VAV systems modulating airflow based on occupancy (maintaining minimum flow when unoccupied).

Once the parameters of each scenario were clearly defined, specific modifications were introduced to the calibrated BEM in order to represent the new conditions. These operations were performed by manipulating the EnergyPlus IDF file using Python and the Eppy library⁸. In this way, the model was adapted to reproduce the conditions of each scenario as accurately as possible. The modified model was then re-simulated to generate updated KPIs, which were subsequently compared with those of the baseline configuration and with alternative scenarios⁹.

Similarly, the scenario definitions were reflected in the BSM, which was updated to represent the spatial and occupancy changes introduced in each case. This process involved editing the Topologic Skeleton Graph (see Chapter 6) – used as the geometric input for safety analysis – in Python using the Topology library. The graph was modified to incorporate the altered spatial configuration and occupant distribution of each scenario. Following these adjustments, the safety simulations were re-run to calculate updated KPIs and verify compliance with safety requirements. The new results were then compared with those obtained for the baseline and alternative scenarios¹⁰.

Simulation runs

Once the IDF models had been edited by adjusting the representative parameters for each scenario, many EP simulations were run, equal to the number of scenarios. Each simulation generated an EP result file in CSV format, which was processed to calculate energy-related KPIs:

- The *Utilisation Rate* expresses how often a given space is occupied relative to the total available time. The *Occupancy Rate* measures the average number of occupants in a space as a percentage of its maximum capacity. Together, these two indicators provide insight into how effectively spaces are used, highlighting underused or overcrowded areas.
- The *Energy Need* indicator estimates the ideal thermal and electrical energy required for building operation – including heating, cooling, lighting, ventilation, and equipment use – expressed in tonnes of oil equivalent (TEP)¹¹.
- The *Energy Cost* KPI translates energy demand into monetary values by applying official conversion factors for electricity and gas tariffs. It provides a comparative estimate of how scenario variations might affect operational expenditure, rather than an exact prediction of real utility costs.
- The *CO₂ Emissions due to Energy Use* KPI evaluates the environmental impact of each scenario by converting energy use into equivalent carbon dioxide emissions (kgCO₂eq), using standardised national conversion coefficients.

Similarly, once the BSMs had been edited by modifying the representative parameters of each scenario, several Python simulations equal to the number of scenarios were executed. Each simulation generated a JSON file containing the results of the safety checks for the simulated scenario.

The safety KPIs correspond to those already defined in Chapter 6 at the element, room, storey, or building levels:

- *Width ratios* (element, room, storey, building): express the ratio between the *required* and the *effective* width of each evacuation path or component. Values greater than 1 indicate non-compliance with the minimum width requirement.
- *Length ratios* (element, room, building): represent the ratio between the *required* and *effective* length of evacuation paths or components. For rooms, the ratio is calculated both locally (within the room itself, for informative pur-

poses) and globally, along the shortest evacuation path connecting the room to its designated exit. Values exceeding 1 are considered not compliant.

- *Independent exits* (room): indicate, as a Boolean value, the presence or absence of independent exits.
- *Openings* (room): express, also as Boolean values, whether doors meet the regulatory requirements for opening direction and handle type.

These KPIs can flag potential safety issues when independent exits are lacking, prompting further evaluation based on the specific use and occupancy conditions of the affected spaces.

9.3.5 Data integration

At this stage, the work continued to integrate the large volume of data and models involved in the analysis using KGs. The semantic alignment and graph creation procedures relied on the Federated Ontology Framework (FOF) presented in Chapter 3, which was extended, as discussed in Chapter 6, to host safety-related concepts. The graph construction process used the JSON-LD syntax and the BTwin library, as presented in Chapter 4, to convert the JSON data into a Neo4j graph.

Creation of the Energy Knowledge Graph

The KG structuring started by lining all energy-related models, objects, and data, following these main steps:

- A `bot:Site` object was first created;
- For each building in the site, a `bot:Building` object was created, along with its spatial hierarchy (containing `brick:Energy_Zone`, and `bot:Space` elements);
- A set of `kpi:Scenario` objects, each associated with specific KPIs, was created to represent the full range of modelled scenarios;
- All related `btwin:Document` objects were then created as nodes and connected to this object, including the IFC model and the scenario-specific BEM models, one for each scenario;
- Each IDF models were linked to their corresponding `kpi:Scenario` node;
- Each IDF model was used in EP to run the simulation, after which the resulting CSV file was saved and modeled as a

btwin:Document, connected both to the bot:Building and the corresponding kpi:Scenario;

- For each CSV file and each KPI, a custom Python algorithm aggregated the data from the EP output to compute the KPIs and group them in btwin:KPISet objects (following a procedure similar to that presented in Chapter 8). These KPIs are then linked to both the bot:Building and the corresponding kpi:Scenario.

Through this structure, queries on the KG can retrieve the KPI of a specific building under a specific scenario by following the sequence: bot:Building \rightarrow eko:hasAssociatedObject \rightarrow btwin:KPISet \rightarrow kpi:relatedScenario \rightarrow kpi:Scenario.

The following two figures illustrate the resulting graph for the case study. Figure 118 shows the spatial hierarchy of two case studies selected within the project, constructed by parsing the IDF models in the graph, following the sequence: bot:Site \rightarrow bot:Building \rightarrow bot:Energy_Zone \rightarrow bot:Space. Figure 119 presents instead an overview of a scenario graph generated for one of the project's case studies. In this figure, the two buildings are located within the same bot:Site, each connected to its own models, scenarios, and energy-related KPIs, according to the logic described above.

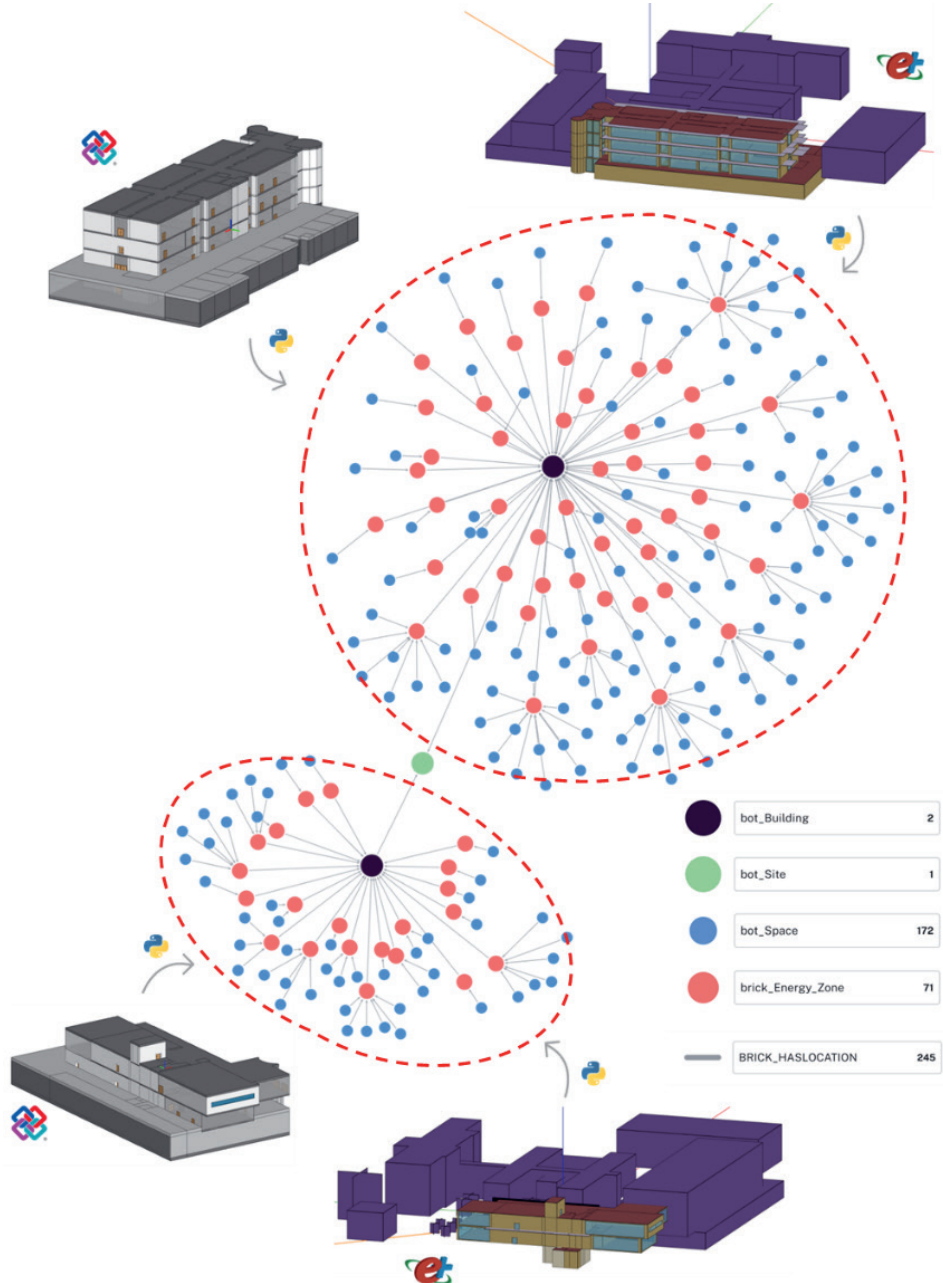


Figure 118: Spatial hierarchy graph for the project's case study.

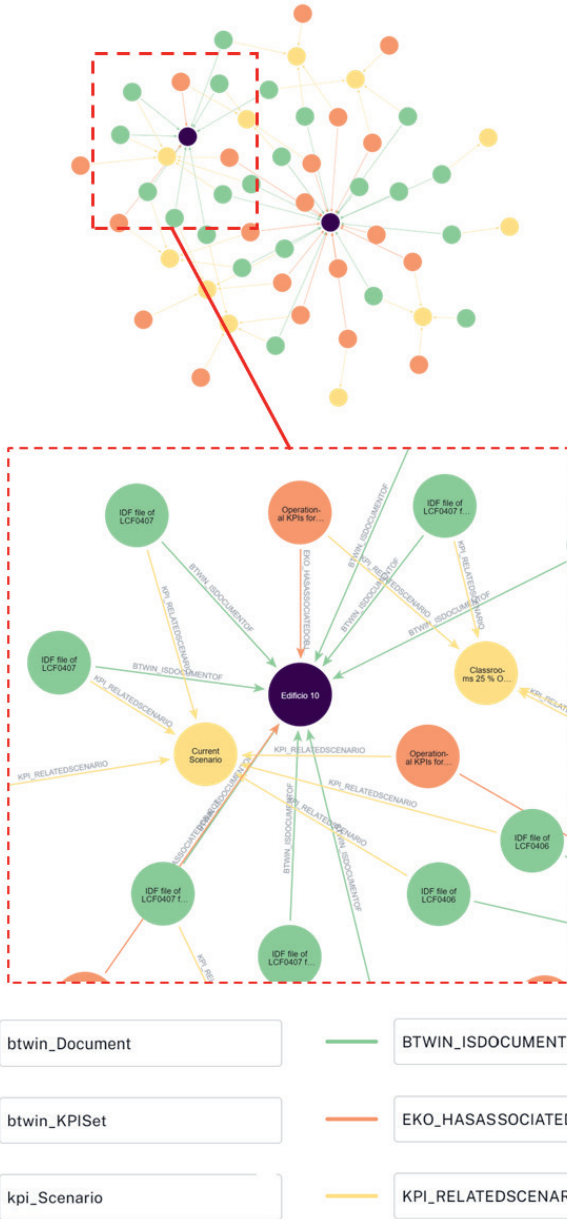
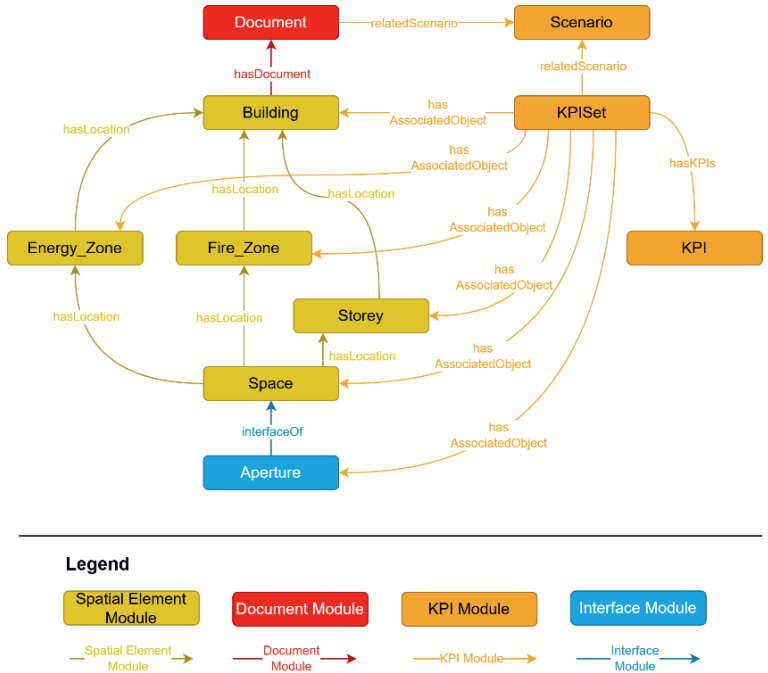


Figure 119: Energy scenario graph for the project's case study.

Integration of safety data in the KG

A second procedure was launched to integrate the extensive data and models involved in the safety analysis into the KG. To achieve this, the structural hierarchy of the BSM model was aligned with that of the BEM model, as illustrated in Figure 120.

Figure 120: Semantic alignment of the spatial structure of the BSM (bot:Building, brick:Fire_Zone, bot:Storey, bot:Space, top:Aperture) with the spatial structure of the BEM (bot:Building, brick:Energy_Zone) and the performance hierarchy.



The workflow proceeded as follows:

- All safety related `btwin:Document` objects were instantiated as nodes and linked to the corresponding `bot:Building` node, previously inserted in the graph, including the Topologic graphs in JSON format representing the current BSM (skeleton and isovist graphs); the NetworkX graphs in JSON format representing the current BSM (passage, skeleton, and isovist graphs); and the scenario-specific graphs (Topologic and NetworkX), one for each scenario;
- Each graph/model was linked to its corresponding `kpi:Scenario` node;

- Each graph was used in Python to run the simulation according to the procedure presented in Chapter 6. The resulting JSON file containing the safety check results was then stored and modelled as a `btwin:Document` node, linked to both the `bot:Building` and the corresponding `kpi:Scenario`;
- For each JSON file and KPI, a custom Python algorithm aggregated the results, linking the `btwin:KPISet` to the corresponding graph element (i.e., `bot:Building`, `brick:Fire_Zone`, `bot:Storey`, `bot:Space`, or `top:Aperture`) and the corresponding `kpi:Scenario`.

As for energy-related metrics, this structure enabled queries on the KG to retrieve the safety KPI of a specific object under a given scenario by following the chain: `bot:Building` → `eko:hasAssociatedObject` → `btwin:KPISet` → `kpi: relatedScenario` → `kpi:Scenario`.

The resulting graph is illustrated in the two figures below. Figure 121 shows the spatial hierarchy of the two case studies considered in the project. The hierarchy is built by aligning the IFC and BSM graphs within the KG and includes the elements `bot:Site`, `bot:Building`, `bot:Storey`, `bot:Space`, `top:Aperture`, and `brick:Fire_Zone`. Figure 122 provides an overview of a scenario graph generated for the case studies, where the two buildings located on the same *site* are shown - together with their fire zones, storeys, and spaces – each connected to its respective models, scenarios, and KPIs according to the logic described above.

9.3.6 Data visualization and microservice development

The KG was employed as a semantic data layer supporting a prototype DSS, which was developed under the DIGITMAN project under a microservice architecture. In particular, the DSS included two modules related to the research here presented: the Energy What-If (EWI) and the Safety What-If (SWI) microservices. Each performs what-if analyses and visualises the resulting data through an interactive dashboard.

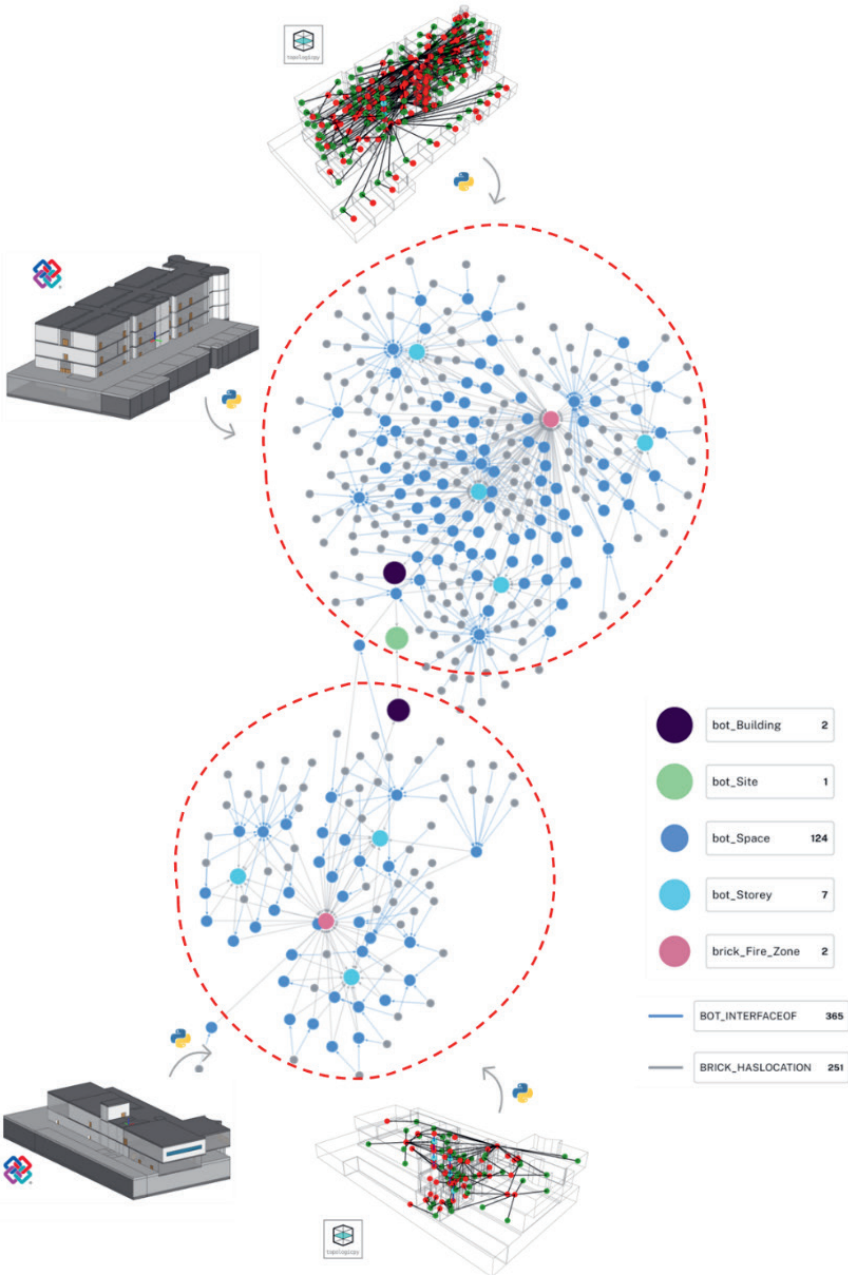


Figure 121: Spatial hierarchy graph for DIGITMAN's case study.

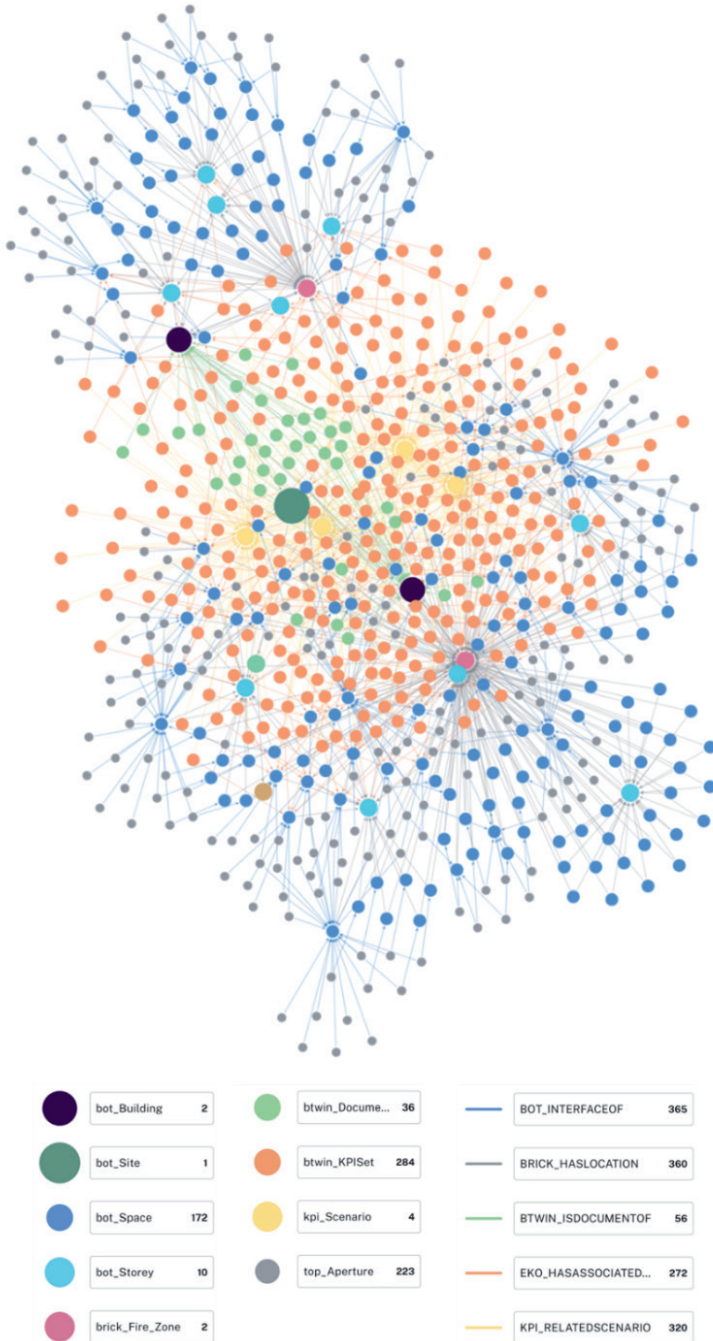


Figure 122: Scenario graph for DIGITMAN's case study.

The Energy What-If microservice

The EWI tool was designed for three key stakeholder groups within the university ecosystem – facility managers, occupancy managers, and building owners – each of whom benefits from its ability to simulate functional changes and assess their energy, economic, and environmental implications¹².

The tool supports occupancy and energy planning by simulating how functional changes affect energy demand for space conditioning, estimating related costs and emissions, and comparing alternative scenarios through the energy-related indicators. It allows decision-makers to evaluate long-term impacts across multiple criteria – occupancy, energy, cost, and emissions – thus supporting building performance management.

In particular, the EWI microservice can:

- Import and manage scenarios involving changes in intended use, occupancy density, occupancy schedules, and energy service modes (e.g. demand-driven controls).
- Analyse baseline and alternative scenarios using a unified KPI set.
- Compare outcomes through quantitative and graphical indicators.
- Generate concise, dashboard-based reports summarising results and insights.

Users access the digital platform, review current building data, and define alternative scenarios to test. The EWI microservice then calculates and compares KPIs between the baseline and alternatives. Finally, a dashboard summarises the results, allowing stakeholders to evaluate the long-term impacts of each configuration and to make informed, data-driven management decisions (Figure 123 and Figure 124).

The Safety What-If microservice

The SWI tool extends the what-if approach to the domain of fire safety. It is targeted at facility, occupancy, and safety managers, as well as building owners and asset managers, supporting them in evaluating whether functional or occupancy changes comply with national fire-safety regulations.

SWI simulates how modifications in building use or occupancy density affect compliance with fire-code parameters related to evacuation systems – namely, the geometric charac-

teristics of escape routes (widths and lengths) and the properties of openings (door direction and handle type). The tool provides an indication of whether each configuration satisfies the ‘deemed-to-satisfy’ criteria defined in the Italian Fire Code. In particular, users can:

- Import and analyse scenarios corresponding to changes in intended use or occupancy density.
- Compare baseline and alternative configurations through safety-related KPIs.
- Generate concise dashboard reports summarising the level of compliance and identifying elements that require further expert evaluation.

The SWI userflow mirrors that of the EWI’s one. After selecting the relevant stakeholder, users define the purpose of the analysis – assessing fire-safety compliance under varying spatial or functional conditions. They visualise the current configuration, define alternatives, and compare KPI outputs across scenarios. The resulting report highlights whether each scenario meets code requirements or requires deeper investigation by fire-safety professionals. This process provides an initial, data-driven understanding of regulatory feasibility before more detailed design assessments are undertaken (Figure 125).

9.4 Results and outcomes

The scenario analysis compared multiple *what-if* configurations to evaluate how variations in ventilation control, space capacity, occupancy schedules, and room function affect the energy, cost, environmental, and safety performance of two pilot buildings. The results of the multicriteria analysis are briefly presented below.

For Building 10, the baseline configuration corresponds to the 2024 lecture hall layout with CAV ventilation. The most significant performance improvements emerged when demand-driven VAV control was introduced. Compared with the baseline, the *VAV-person* configuration reduced heating demand by 24 %, cooling demand by 7 %, total primary energy by 4 %, operational costs by 9 %, and CO₂ emissions by 11 %. In contrast, CAV control based on occupant count increased energy

Figure 123. Dashboard of the EWI microservice. The upper panel provides filtering options for selecting the building and scenario. The lower section displays a 3D view of the spaces, colour-coded according to the relevant KPIs. On the right, numerical comparisons and a radar chart illustrate the differences across the various scenarios.

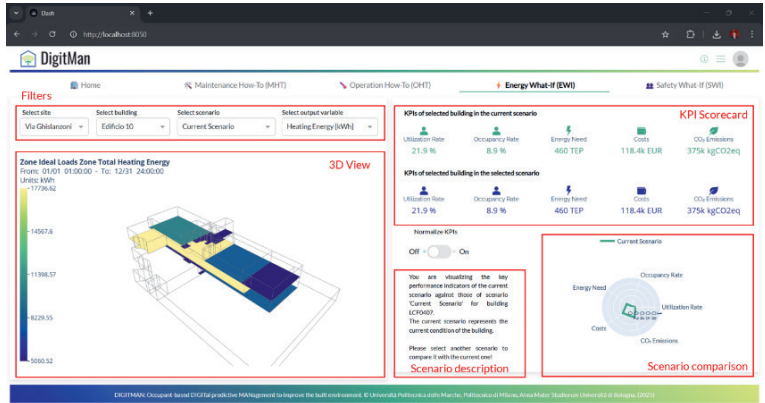


Figure 124. Comparison between the current scenario and the 'Classroom CAV-person' scenario for Building 10.

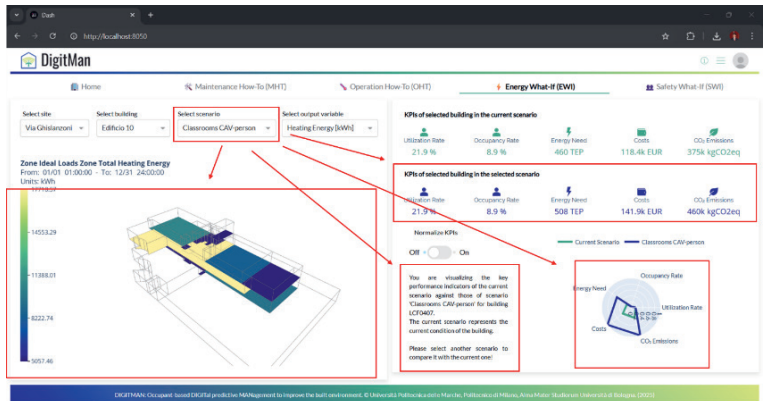
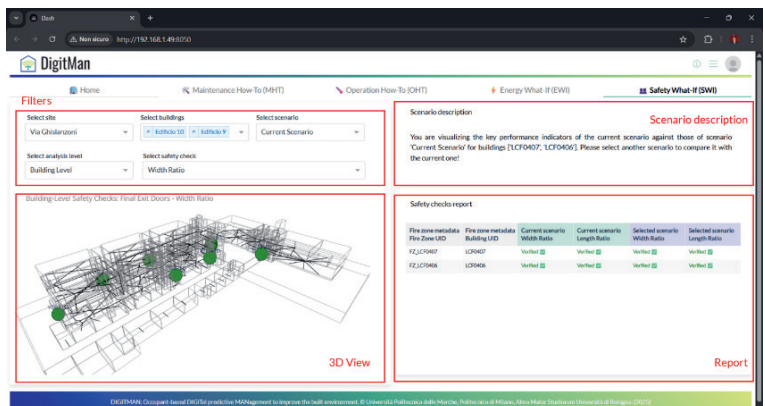


Figure 125. Dashboard of the SWI microservice. The upper panel enables the selection of both the scenario and the KPIs. The lower section presents a 3D view of the building. On the right, a description of the scenario and the corresponding analysis is provided, together with a tabular report.



demand and costs. Changes in nominal seating capacity ($\pm 20\%$) produced only minor effects under volume-based ventilation. Adjusting occupancy schedules revealed that higher utilisation led to strong increases in summer cooling demand, while reduced occupancy resulted in near-baseline conditions. Repurposing the lecture halls as closed offices under CAV operation increased energy needs, but coupling the same configuration with demand-driven VAV control reversed the trend, cutting emissions by up to 17%. Overall, Building 10 proved highly sensitive to occupancy and ventilation strategy, confirming that demand-controlled VAV systems offer the most effective energy and environmental performance.

For Building 09, which shares a similar configuration but features a longitudinal south-facing façade, solar gains dominate its energy balance, yielding 35% higher cooling demand per square metre than Building 10. Scenario outcomes followed the same trend: the VAV-*person* configuration offered the best performance, reducing heating energy by 46%, cooling energy by 25%, and lowering TEP, costs, and CO₂ emissions by 13%, 19%, and 21%, respectively. Capacity and occupancy schedule variations produced limited effects due to the stronger influence of solar loads. As in Building 10, converting lecture halls into offices under CAV control increased energy use, while introducing demand-based VAV ventilation achieved up to 18% emission reductions.

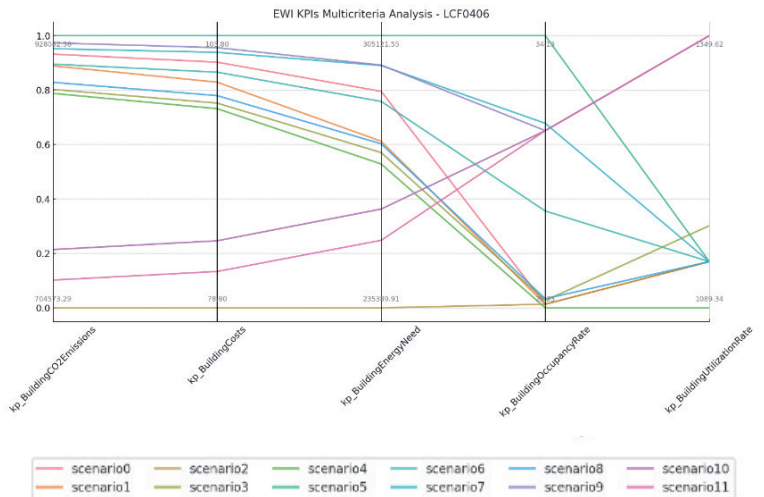
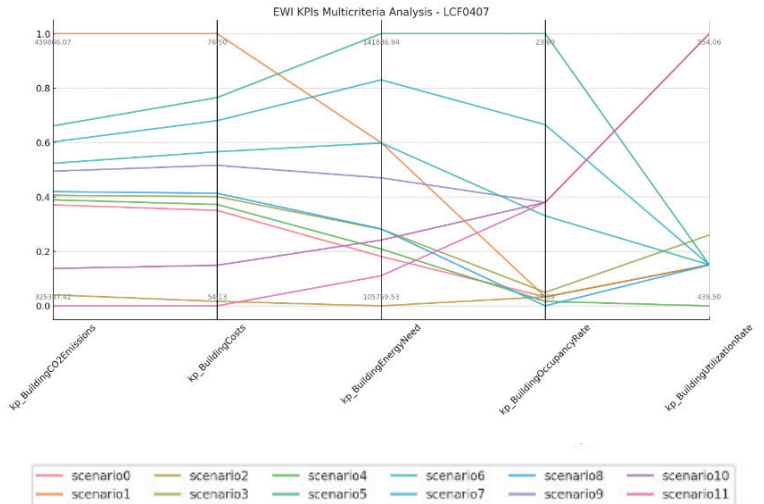
The *what-if* safety assessment further examined compliance with national fire regulations across all scenarios. The analysis shows that every configuration, except Scenario 3 (Classrooms +20% capacity), meets code requirements. The baseline condition for 2024 lecture halls (Scenario 0) is fully compliant, as expected, since it was designed according to National Fire Regulations. Lower-density scenarios, such as Scenario 4 (Classrooms -20% capacity) and Scenario 9 (Closed offices without public access), are likewise compliant, as their reduced occupant loads maintain adequate egress widths and compartmentation.

By contrast, Scenario 3 triggers a safety alert: increasing the nominal headcount by 20% exceeds the capacity of certain evacuation routes, whose exit widths are insufficient to accommodate the higher density. While a few rooms could technically sustain the additional occupancy, this condition cannot be gen-

Figure 128. Safety analysis for the demonstration building 'Building 09' and 'Building 10' – Building Level Checks – Width Ratio. Scenario 0 (top) vs Scenario 3 (bottom).

eralised without architectural modifications to the evacuation system.

Stakeholders are thus immediately informed that Scenario 3 should either (1) be excluded at the EWI decision stage when considering generalised occupancy increases; (2) be adapted by



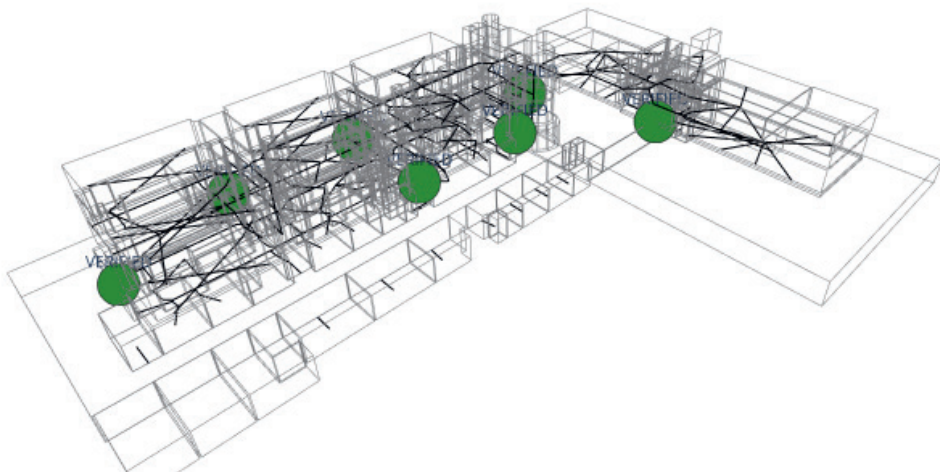


Figure 126. Parallel coordinate plot showing the energy-related KPIs for the demonstration building 'Building 10'.

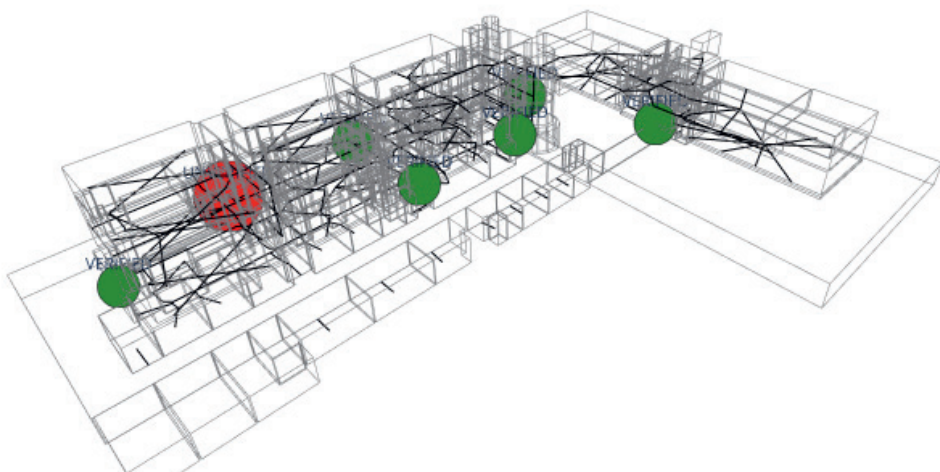


Figure 127. Parallel coordinate plot showing the energy-related KPIs for the demonstration building 'Building 09'.

Figure 129. Safety analysis for the demonstration building 'Building 09' and 'Building 10' - Storey Level Checks - Width

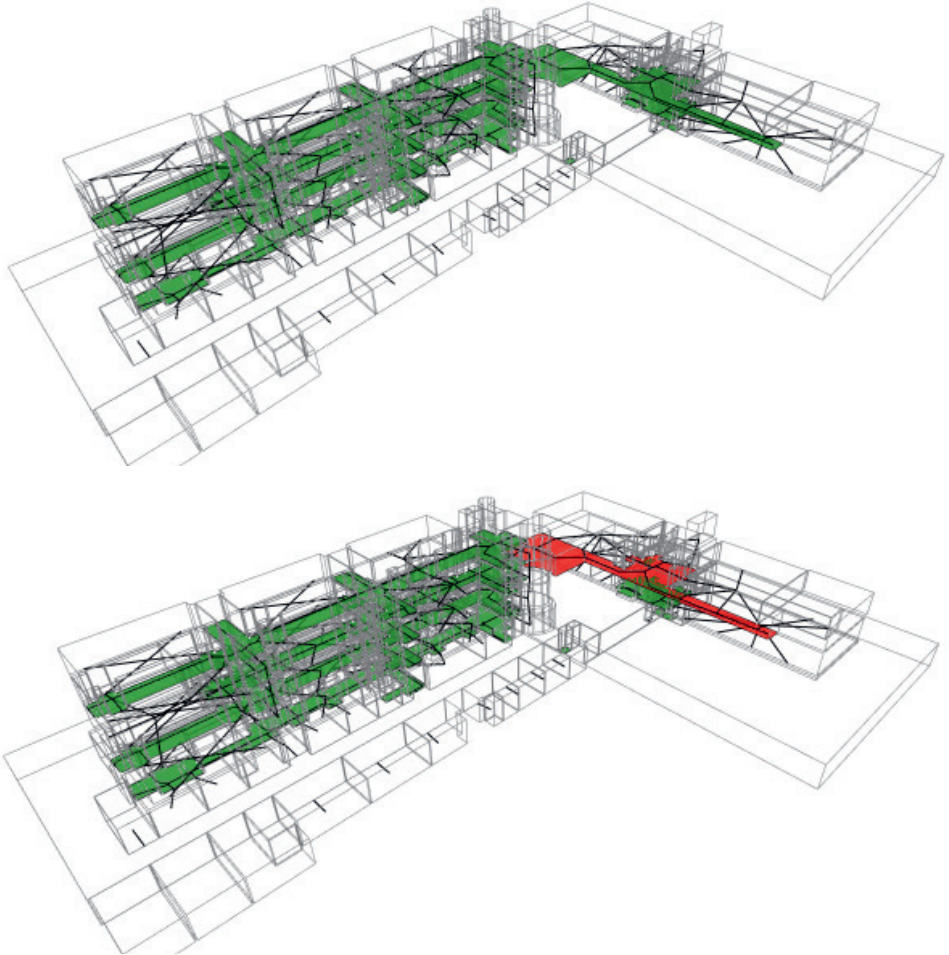


Figure 130. Safety analysis for the demonstration building 'Building 09' and 'Building 10' – Room Level Checks – With Ratio. Scenario 0 (top) vs Scenario 3 (bottom).

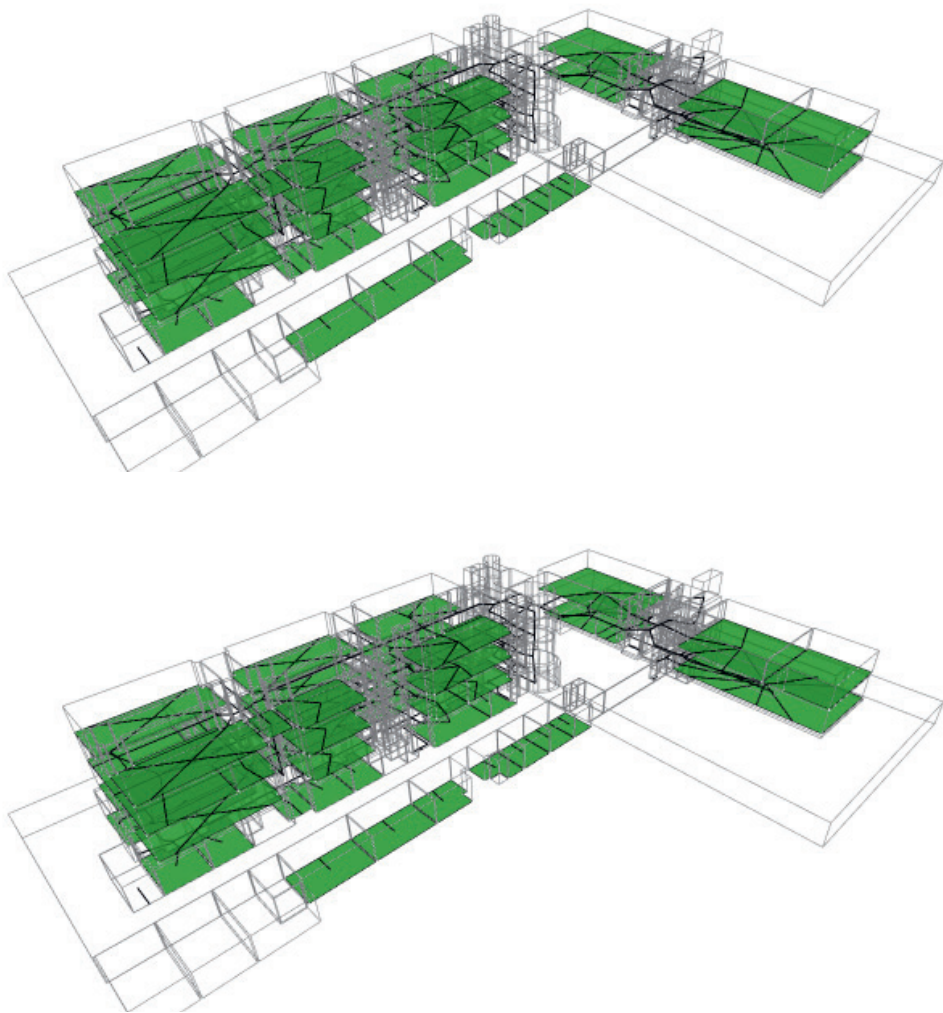
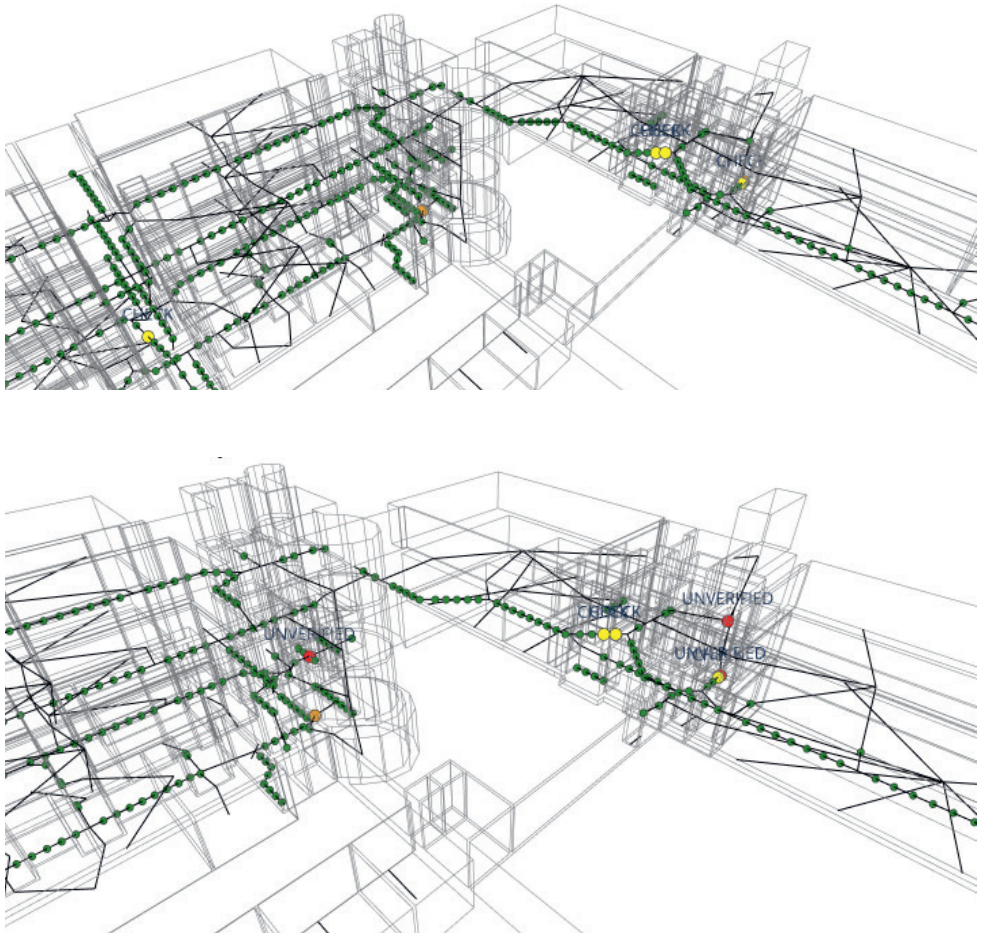


Figure 131. Safety analysis for the demonstration building 'Building 09' and 'Building 10' – Element Level Checks – Horizontal Elements With Ratio. Scenario 0 (top) vs Scenario 3 (bottom).



Notes

¹ The zoning procedure combined topological methods – used to group adjacent cells – with machine learning techniques, specifically the K-Means clustering algorithm, which further clusters spaces based on their occupancy and operational characteristics extracted from the IFC model. For this case study, the attributes considered for space clustering included “IsOccupied”, “IsHeated”, “IsCooled”, and “AreaPerOccupant”.

² Flow rate per exterior surface area was set equal to $0.0001 \text{ m}^3/\text{s}\cdot\text{m}^2$, referring to a tight building for the case study in question.

³ For occupancy schedules, which were particularly significant for the project, data directly from the university’s asset management service were considered.

⁴ Given the homogeneity of the buildings’ construction systems, these classrooms were considered representative of the whole building. For both the classrooms, a dataset was assembled containing hourly records for an entire year (8,760 data points), including timestamps, real temperatures measured by wall-mounted sensors, and simulated temperatures obtained from EnergyPlus ‘Zone Mean Radiant Temperature’ outputs. Sensor and climate data covered the same period (February 2024–January 2025) to ensure temporal consistency.

⁵ Root Mean Square Error (RMSE) is a standard metric for quantifying the differences between values predicted by a model and the actual observed values. It is calculated as the square root of the average of the squared errors (residuals) and indicates the typical magnitude of the error in a set of predictions. A lower RMSE value means the model’s predictions are closer to the actual values, indicating higher accuracy.

⁶ A ‘what-if’ scenario refers to a hypothetical situation analysed to evaluate the potential outcomes of alternative decisions or system changes. This approach is widely employed in project management, strategic planning, and performance modelling to assess risks, identify opportunities, and anticipate the consequences of specific interventions.

⁷ Each of these questions entails adjustments to occupancy-related parameters that, as extensively demonstrated in the literature (Gui et al. 2021; Matsunobu and Coimbra 2023; Mosteiro-Romero et al. 2023), can substantially influence both the energy and safety performance of buildings.

⁸ In technical terms, four main types of modifications were applied. First, changes in space capacity were implemented by adjusting the **People per Floor Area** property of the **People** object corresponding to the affected zone, thereby modifying the occupant density. Second, when occupancy schedules were altered, a new schedule was created and assigned to the **Number of People Schedule Name** property of the **People** object. At the same time, the **Schedule Name** property of the **ElectricEquipment** object was updated, assuming that plug-load consumption is directly proportional to occupancy levels. Third, in cases of space repurposing, the same procedure as for capacity variation was applied: the **People per Floor Area** property of the **People** object was edited to reflect the new intended use of the space.

⁹ Finally, demand-driven ventilation strategies were simulated by modifying the **DesignSpecification:OutdoorAir** object for each zone. For CAV systems, the **Outdoor_Air_Method** was set to **Flow/Area** for systems based on room size and to **Flow/Person** for systems based on maximum oc-

cupancy. In the case of VAV systems, the same `Outdoor_Air_Method` was used, but the `Outdoor_Air_Schedule_Name` property was linked to the occupancy schedule of the `People` object, which contained hourly values ranging from 0 (no occupants) to 1 (maximum occupancy). During unoccupied periods, a minimum ventilation rate was maintained to replicate real operating conditions better.

¹⁰ In practical terms, modifications to the BSM primarily concerned the `Occupancy Density` property stored within the `Topologic Dictionary` associated with each `Vertex` in the Skeleton graph. For example, increasing a classroom's capacity by 20% required increasing its occupancy density from 1.20 to 1.44 persons per square metre. In cases involving changes in intended use, the same parameter was adjusted to reflect the new function: converting a classroom into an office, for instance, meant reducing the occupancy density from 1.20 to 0.40 persons per square metre, thereby reflecting the different spatial and functional requirements.

¹¹ Since HVAC systems were modelled in a simplified way in the BEM, although produced through a calibrated model, this metric does not represent actual consumption but rather the theoretical energy demand.

¹² Facility managers, including energy managers and those responsible for specific building systems, can use EWI to assess the operational impact of different configurations on energy consumption, costs, and emissions, thereby supporting evidence-based decision-making aligned with institutional energy-performance goals. Occupancy managers, such as departmental coordinators and scheduling staff, can visualise how changes in space use patterns influence energy demand, enabling more efficient planning and avoidance of under- or over-utilisation of spaces. Building owners or asset managers can rely on EWI for strategic planning, comparing the potential benefits of operational adjustments with those of retrofit interventions, and prioritising actions based on quantified impacts across the building portfolio.

¹³ These last options should be reviewed in consultation with fire safety experts to ensure regulatory compliance, but the general modelling approach remains valid and can rapidly visualise the outcomes through KPI-based analysis. In such cases, the BIM – and consequently the topologic model – can be locally updated at the component scale to reflect revised spatial and occupancy conditions.

FUTURE SCENARIOS

10. TOWARDS AI-INTEGRATED KNOWLEDGE SYSTEMS

Abstract

The convergence of graph-based knowledge modelling and Artificial Intelligence (AI) is opening new frontiers for the AECO sector. This emerging field of knowledge lies at the heart of a profound transitional era that is influencing every area of expertise and all human activities, including the construction sector. The core of this paradigm shift, typical of scientific revolutions, does not rest solely on the technological aspects of the issue, but rather on the broader challenge of fostering awareness in decision-making, from which the question of ethical responsibility naturally follows.

This chapter explores how Knowledge Graphs (KGs) – previously used to integrate heterogeneous building data – can be coupled with Large Language Models (LLMs) to create AI-integrated knowledge systems. It discusses the emergence of natural language interfaces for interacting with complex building data, the challenges of ensuring that AI-provided information remains valid and traceable, and how KG technologies can complement LLMs by grounding them in factual domain data. Then, the text outlines example architectures for LLM-KG integration and demonstrates a fictional application using Claude (an LLM) connected to a Neo4j-based building KG to support natural language queries from the graph.

Future scenarios envision AI assistants, guided by domain-specific ontologies and validation standards, becoming trusted partners in building design and management.

10.1 From graph thinking to AI reasoning

As observed in the first chapters of this book, over the past decade *graph thinking* – the practice of modelling knowledge as networks of entities and relationships – has gained prominence as a means of tackling the complexity of building information, usually dispersed across different subdomains of the AECO sector. In this field, graph data structures are widely used today, especially within the research community, to represent knowledge of buildings, their components, and their processes throughout the lifecycle. In particular, KGs have proved highly effective for integrating diverse building datasets into a unified, queryable format, as demonstrated by the prototypical applications presented in the previous chapters of this book.

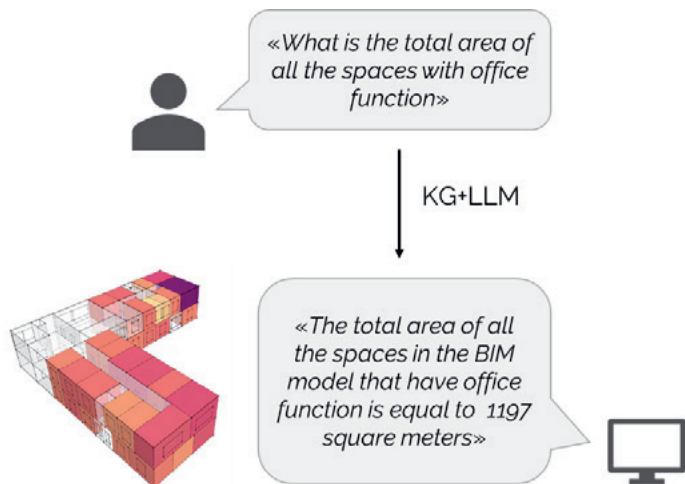
Unlike raw data silos, KGs embed semantic relationships defined by ontologies, making the data both human-interpretable and machine-understandable. This semantic enrichment allows building knowledge to be organised in a way that is not only comprehensive but also easily queryable. To address the problem of data accessibility for non-digital experts, some of the applications presented above have coupled KGs with web-based *business intelligence* solutions, such as interactive dashboards for visualising key indicators, thereby supporting data-driven decision making.

However, things are changing – and also very rapidly. As the reader will know, the past three years have been transformative for research in digital AECO (Memon *et al.* 2025) and, indeed, for all disciplines that in some way concern the digital modelling of knowledge. This revolution has not been driven by graphs – which, though conceptually fascinating, are rarely attractive outside academic circles – but rather by Generative Artificial Intelligence (GenAI), which has indubitably taken centre stage. This revolution first introduced LLMs – advanced AI systems that rely on deep learning and vast datasets to understand, generate, and process human language – and, more recently, AI agents, including multimodal ones – these are software systems that leverage AI to pursue goals and carry out tasks on behalf of users, demonstrating reasoning, planning, and memory capabilities, and possessing a degree of autonomy that enables them to make decisions, learn, and adapt. This innovative technology

is bringing about a genuinely disruptive revolution, in the Popperian sense of the term: a paradigm shift in the foundations of scientific method (Popper and Popper 2008).

It is now unavoidable to acknowledge that, despite the many problems associated with AI, its potential exceeds what could have been imagined only a few years ago. It therefore seems inevitable that the next frontier in knowledge management will consist of integrating already established knowledge technologies, such as KGs, with AI capabilities – an association that, according to recent experiments reported in the literature, appears to work remarkably well (Pan *et al.* 2023) such as ChatGPT and GPT4, are making new waves in the field of natural language processing and artificial intelligence, due to their emergent ability and generalizability. However, LLMs are black-box models, which often fall short of capturing and accessing factual knowledge. In contrast, Knowledge Graphs (KGs). Recent advances in AI, particularly LLMs, suggest that natural language can become a powerful interface for querying and manipulating complex graph data. More importantly, using KGs as knowledge bases to support LLMs significantly reduces the so-called ‘hallucinations’ these models are prone to. In other words, the advantages of the two technologies converge: the semantic and contextual structuring of knowledge offered by KGs, and the human-centred interaction provided by LLMs.

Figure 132. User-machine interaction through KG and LLMs.



At the time of conceiving this book, I was aware – or perhaps apprehensive – that by the time the manuscript was completed, some of its content might already appear technologically obsolete. This, of course, is the result of the relentless pace of progress in IT technologies, which now achieves in weeks what once required years or even decades of research. This is precisely why, as I have repeatedly tried to explain, my primary goal in this work has been to convey an approach – a methodology for knowledge modelling – intended to bring order and structure to my thoughts over the past years and to codify them within the notion of *Buildings as Networks*, whose conceptual roots are by no means my own invention.

Throughout the book, I have repeatedly highlighted how federated ontologies and graph databases can serve as a ‘semantic backbone’ for building information and related knowledge systems. This is now, for me, a consolidated technique in the research field, although it is still rare in professional practice for a variety of reasons, both cultural and technical.

This chapter, therefore, makes a tentative effort to extend that vision towards the future, and towards what will inevitably be the next phase of this research in the coming years, shifting attention to the theme of AI integration in knowledge systems. Accordingly, the central question of this last section becomes: *How can graph-based knowledge systems and AI work together to improve human-machine collaboration in AECO?* It would be premature to attempt a complete answer here – dozens of researchers worldwide are working on it. Instead, this chapter attempts to offer an anticipation of the future that awaits us, accompanied – as is my habit – by a few small practical examples to help the reader grasp concepts that might otherwise appear abstract, over-ambitious, or technically ungrounded.

An optimistic perspective is that this work – conceived for people, particularly those working in the field of building knowledge modelling and students approaching this discipline – will also serve machines. Ideally, the knowledge and conceptual structure proposed here will enable even non-expert users to interact with machines without requiring any digital skills, simply by communicating with them through natural language. After all, this is the most desirable form of digitalisation we can aspire to: one that simplifies communication.

10.2 The rise of natural language interfaces in AECO

Over the past two years, *natural language interfaces* have emerged as highly promising digital environments for professionals to access and interact with building information. Traditionally, extracting insights from a building information model or a performance database required specialised expertise. One needed to know which software to use (for example, IFC viewers or BIM tools to visualise models) or which query language to employ (such as SQL, SPARQL, or Cypher), often requiring the learning of programming languages – both textual and visual – for parsing data and models. This technical barrier frequently left vast quantities of valuable data under-utilised.

In contrast, LLMs offer a way to *converse* with machines and software, posing questions in plain English (or any human language) and receiving meaningful – or, apparently meaningful – answers. Of course, we must acknowledge the many limitations that still characterise this technology, both ethical and hermetic, but its potential cannot be denied. These are basically systems that understand – or seem to understand – both questions and context before formulating their responses.

In the AECO context, where one of the major barriers to the diffusion of a key knowledge technology such as BIM lies precisely in the limited accessibility experienced by small- and medium-scale operators, this innovation is far from trivial. Thanks to AI, natural interaction with BIM, digital twins, or any other knowledge system now appears closer than ever, and the barriers to using such systems have never seemed so low. We are probably leaving behind a world of software made up of functions and buttons, moving towards one based on chat and dialogue. This is a very recent field, still largely unexplored except by a few pioneers, and evolving literally day by day.

In any case, the appeal of such interfaces has not gone unnoticed – both within and beyond AECO (Lee *et al.* 2024) it is foreseeable that BIM tasks, including querying and managing BIM data, 4D and 5D BIM, design compliance checking, or authoring a design, using written or spoken natural language (i.e., text-to-BIM or speech-to-BIM. AI has, in effect, been democratised – or, more accurately, its use has been, since the technology itself remains the property of a few private companies. People of

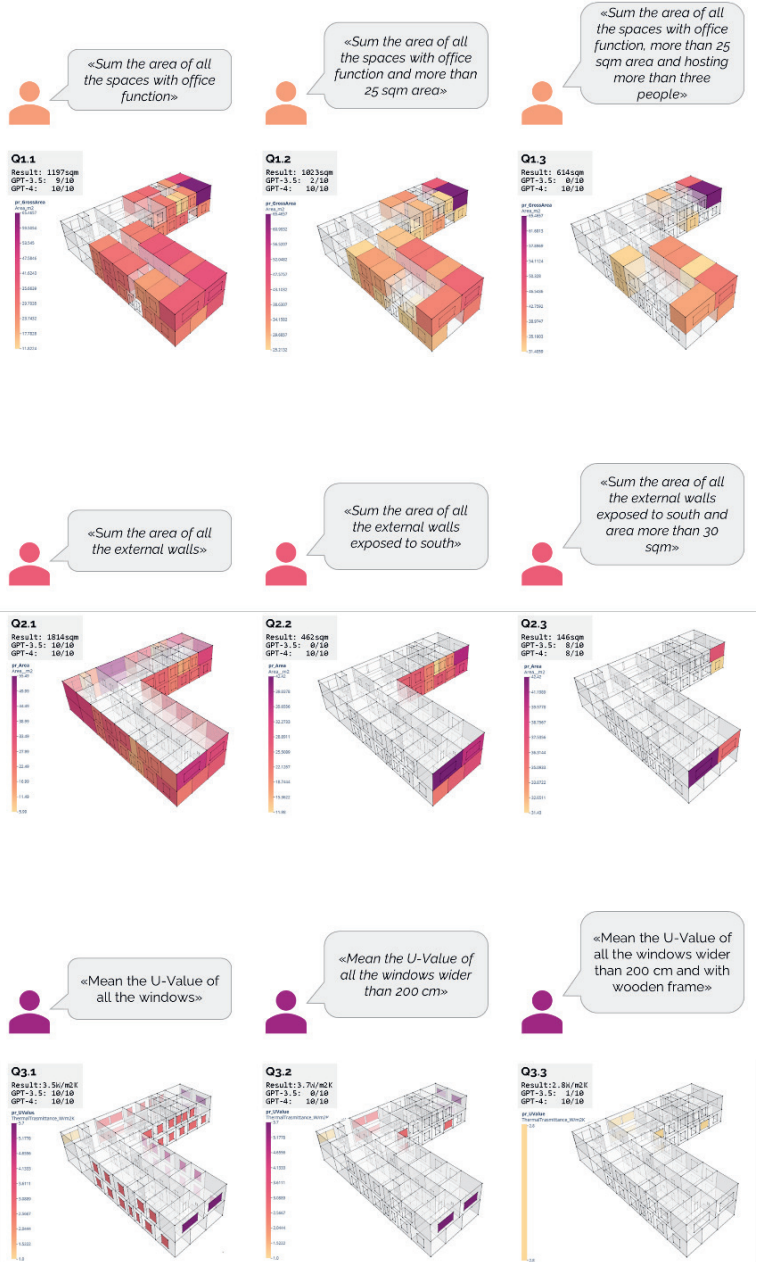
all educational backgrounds have begun using ChatGPT and its counterparts for a wide range of purposes.

Within AECO research, scholars have increasingly begun exploring integrations between BIM and conversational AI. The literature is constantly evolving, and closely follows the capabilities of the latest LLMs released almost daily. One of the earliest attempts probably comes from Rane et al. (2023) (Rane et al. 2023), who proposed a framework in which "*BIM's collaborative ethos is extended through natural language interfaces from ChatGPT, fostering seamless communication and idea exchange among project stakeholders*". Early industrial prototypes already hint at these possibilities, with companies beginning to offer AI-driven chatbots for project management and building analytics.

I myself, together with my research group, reflected in the 'distant' 2023 on how AI could be used for knowledge retrieval from BIM models, presenting a year later, in 2024, the paper *Building Information Model Analysis Through Large Language Models and Knowledge Graphs* at the eCAADe 2024 conference (Figure 133). I recall that at the time of writing, one year before publication, it was almost impossible to find references in the literature – not because no one was working on it, but because the publication cycles of academic journals are commonly slow compared with the speed of digital technological development. Today, only two years later, the rapid evolution of AI systems already makes that research appear like a 'primitive relic of the digital Jurassic' – especially when compared to the further advances that will occur between the moment of writing this chapter and the eventual publication of this book.

On this subject, one could open a broader reflection on the role of architectural researchers in a context so dependent on the advancement of a single technology. Yet, as this is a technical book, such a discussion – though fascinating – would risk becoming inconclusive and moving beyond the aims of the present work. Leaving aside the Asimovian and, sometimes, apocalyptic scenarios that we all, in one way or another, fear or imagine, as well as the philosophical discussions on the so-called 'Age of Technology' and the corresponding 'Ethics of the Wanderer' proposed by Galimberti (Galimberti 2023) – according to which, in the technological age, technology becomes an

Figure 133. Extract from the presentation given at the eCAADe 2024 conference about 'Building Information Model Analysis Through Large Language Models and Knowledge Graphs' (Massafra et al. 2024b).

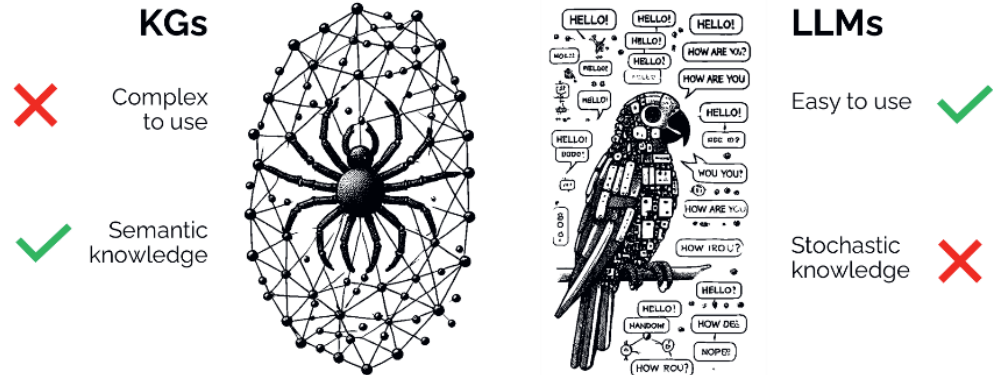


end in itself rather than a means for humanity, leaving us to wander equipped with immensely powerful tools but without a clear destination – what matters here is the pragmatic recognition that LLMs and AI agents can indeed lower the technical barriers to data accessibility – and can do so now, simply and effectively.

In my view, this is precisely what should most concern the construction sector at this historical juncture, a field plagued by a level of knowledge fragmentation found in few other domains – where expertise and languages belong to actors who, due to various personal and professional biases, communicate in highly divergent ways. This is not due to individual deficiencies, but to the intrinsic nature of a sector that is profoundly multidisciplinary and, at times, chaotic. In other words, if properly guided, AI could significantly reduce the knowledge gap that affects the AECO sector.

Natural language interfaces promise to make advanced decision-support tools more user-friendly, accessible, and inclusive – simply because they compute as fast as machines but communicate as effectively as humans, in discursive, not necessarily structured, form. However – and this is, at present, the actual limitation – providing natural-language access to critical knowledge raises one fundamental concern: *can we trust the answers given by AI?* This leads us directly to the next section, which examines the challenge of maintaining valid and reliable knowledge in human-machine dialogues.

Figure 134. The Deterministic Spider (KG) vs the Stochastic Parrot (LLM).



10.3 Challenges of valid knowledge in human–machine dialogue

While AI chat interfaces appear to be extraordinarily powerful tools – capable of things unimaginable only a few years ago – they bring with them a significant caveat: the *validity of the knowledge* they provide. LLMs are known to sometimes produce information that sounds plausible but is in fact incorrect or unsubstantiated – a phenomenon often referred to as *hallucination*. In a casual context, a spurious answer from an AI assistant may be merely an annoyance; in the AECO domain, however, decisions based on incorrect information can have serious implications. To give a simple yet immediate example, an AI that improvises a structural load value for a beam or misinterprets a fire safety regulation could easily mislead a project team if its output is not correctly verified and validated through specialist methods or software.

Several factors contribute to this challenge of ensuring valid answers. First, general-purpose LLMs – even those trained with billions of parameters – still lack guaranteed accuracy in specialised domains. This lack of precision increases as one descends vertically into a specific field or narrows the focus to a local scale.

To illustrate, this year, with my students in the *Building Renovation Workshop*, we asked ourselves, failing, whether AI systems could recognise the construction characteristics of certain recurring building types in the Bologna area, to assess whether AI could assist in understanding the built heritage – particularly for students who still have limited experience in identifying local materials and construction techniques. I also recall a design review in which one of my students asked ChatGPT to design a structural improvement to the junction between a wall and a floor slab in a masonry building. While the conceptual design framework provided by ChatGPT was sufficient to justify the project's overall choices, the technological details were entirely inadequate. Beyond the evident graphic deficiency – which is not, after all, among the major tasks of a language model – the real issue lies in the fact that general-purpose AI does not 'like' the scale of local detail, at least not yet. This is likely because such systems learn from broad web texts, which rarely include

the precise technical standards or up-to-date building codes relevant to a specific query. Without integration into a trusted knowledge base, the AI fills gaps with its own ‘best guess’ – and that guess may be wrong. Second, LLMs have no inherent mechanism for showing the *source* of their answers – or at least, not always. Unlike a human expert, who can cite a code clause, a standard, or a measurement report, an LLM typically does not provide justification unless it is explicitly engineered to do so (for example, through retrieval-augmented generation or by using specific prompts that request citations).

From the perspective of knowledge modelling, this marks a clear departure from the rigour that KGs and ontologies strive to maintain. As we have seen in previous chapters, a well-crafted KG in the building domain encodes factually vetted relationships and can be linked to provenance information. If an AI bypasses that and relies solely on its internal ‘training memory’, the *scientific validity* of its information may be compromised. It is therefore undeniable that researchers play a crucial role in ensuring that digital decision-support systems in AECO enable *reliable and scientifically valid* information generation. Achieving this requires careful design when introducing AI into the loop – not merely for developing prototypes or visually appealing renderings, but for actual deployment within operational automation processes.

Fortunately, research is already beginning to address this issue. One immediate mitigation strategy is *grounding* the AI’s responses in a vetted knowledge source. This is precisely the logic behind RAG techniques, in which the AI is not left to generate answers from its general knowledge but is compelled to draw from external data – documents, databases, or KGs. Another challenge is maintaining *context* and *constraints*; in other words, in a multi-turn dialogue, how can we ensure that an AI’s follow-up answers remain consistent with previously retrieved facts and adhere to domain rules? This becomes an engineering problem about aligning LLMs with KGs, a topic explored in the following paragraph.

Beneath all these technical solutions, however, lies a broader truth: valid knowledge in human–machine dialogue is not totally automatic; it must be actively enforced through system design. Beyond this, human oversight remains the most effective

safeguard against hallucinations, wherever it can be applied. From a pedagogical perspective, this raises an essential point: we should *educate ourselves* to use these new tools with seriousness and awareness. Critical thinking remains firmly within the domain of the human mind, not the machine. We should resist the temptation to delegate our judgement to algorithms – and instead treat AI as an ally that extends, rather than replaces, our capacity for reasoning.

10.4 Complementarity between the two technologies

Despite the challenges discussed so far, there is a growing consensus that KGs and LLMs complement each other's strengths.

On the one hand, KGs excel at providing a structured, context-rich representation of domain knowledge. They store facts with precise relationships and can be queried with exact logic – for instance, «*find all sensors located in rooms on the second floor that measure temperature.*» KGs are *reliable by construction*, by their very nature, since they typically aggregate curated data and enforce ontological consistency. However, interacting with a KG usually requires significant technical expertise, such as the ability to write SPARQL or Cypher queries or understand the graph schema.

LLMs, on the other hand, are masters of flexibility and abstraction – so much so that at times they seem to invent new things or concepts. They can interpret imprecise or incomplete questions, decompose complex linguistic input, and generate coherent, contextually appropriate responses in natural language. LLMs can also perform a form of *soft reasoning*, synthesising information and even making plausible inferences or suggestions. What they lack, however, is assured correctness and domain grounding. In fact, all the *emergent capabilities* of LLMs derive essentially from concepts in statistics, probability, and stochastic modelling, which, when pushed to extremes, seem to constitute a form of intelligence.

This is where KGs come in. By connecting an LLM to a KG, we provide the model with a *source of truth* from which to draw, rather than relying solely on its parametric memory. In other words, the KG provides knowledge, context, and semantics,

while the LLM provides eloquence, interpretation, and communication.

We can imagine this dichotomy through a metaphor illustrated in Figure 1: the *Deterministic Spider* (the KG) and the *Stochastic Parrot* (the LLM). The spider is a serious and rigorous creature. It builds its web with care, precision, and an engineering knowledge that only it possesses. The web has a specific structure: its threads have a given thickness and tension, and their connections follow a logic that the spider instinctively knows to be necessary. Nothing is left to chance. Yet the spider is not flashy, and its work – though vital to the ecosystem’s food chain – often goes unnoticed. In other words, the spider lacks an effective marketing strategy. And yet, without that meticulous understanding of matter and its relationships, the web – arguably an example of structural engineering and, in a sense, parametric design – would not function. The stochastic parrot is its opposite. It is immensely powerful because it has listened to billions of words and can repeat them, often in the proper order, appearing even intelligent. However, the parrot repeats only what it has heard – the *sound*, not the *meaning*; it does not know the context and has never seen it. Despite its flamboyant plumage, when asked a question requiring inference or deep reasoning, it tends to provide generic answers that soon make its interlocutor doubt that it truly *knows* anything. Its marketing and communicative appeal – apparently enormous – collapses the moment it is required to take things seriously and demonstrate reliability.

Not wanting to drift too far into a Phaedrus fable, let me give a more practical example from the building sector to illustrate the complementarity between KGs and LLMs in RAG scenarios.

Suppose a user asks: «*What is the typical thermal conductivity of the material used in the external walls of Building X?*» A KG might store the relevant material properties – perhaps linked to an ontology such as IFC or to an external materials database. By working together, the LLM can identify the intent of the question and the relevant entity (external wall, Building X, material property), then retrieve the value from the KG, and finally phrase the answer in an explanatory way: «*The external walls of Building X are made of brick masonry, which has a thermal conductivity of 0.72 W/mK according to the building’s material database.*»

The *LLM+KG duo* thus delivers an answer that is both articulate and grounded in factual data, with the KG ensuring traceability of the information source (that 0.72 W/m·K). The semantic structure of the KG helps the LLM navigate the hierarchy of elements, determining which objects to inspect and which properties to read, selecting within a predefined schema of the expected answer field.

Beyond simple question–answer interactions, KGs can also guide LLM reasoning by imposing *logical constraints*. For example, if a user’s query requires comparing data across different building systems, the ontology underlying the KG can clarify relationships (e.g. HVAC zones belong to floors; sensors belong to zones), ensuring that the LLM’s analysis respects the actual building structure. Conversely, the LLM can assist by bridging *disparate vocabularies* – translating a user’s informal question, such as «*Are there any rooms that are energy hogs?*», into a formal graph query: ‘*rooms with unusually high energy use per area*’.

This synergy shows that AI-integrated systems can indeed harness the best of both worlds: the precision and rigour of symbolic knowledge representations and the adaptability and expressiveness of statistical language understanding.

10.5 Architectures for LLM–KG Integration

Marrying an LLM with a KG requires a thoughtful and carefully designed system architecture. Various architectures are currently under development. What follows is a deliberately concise and non-exhaustive overview. For a deeper treatment of the topic, readers may refer to works such as those by Kau et al. (Kau et al. 2024) and Škrlj et al. (Škrlj et al. 2025)

A widely adopted approach is RAG, in which the LLM is enhanced by a retrieval component that fetches relevant data from the KG (or associated databases) and incorporates it into the model’s prompt. In a graph-based RAG pipeline, a user query might first be translated into a graph query (for example, in Cypher or SPARQL) or into a semantic search across the KG. The top results – facts, data points, or short text descriptions – are then fed into the LLM’s context window, so that the model’s eventual response is *augmented* with real data.

Studies have shown that this method can significantly improve the factual accuracy of LLM outputs in specialised domains, as the model is less likely to hallucinate when it can rely on domain-specific evidence. For instance, *SparqLLM* (Arazzi *et al.* 2025) is a framework that uses a RAG approach to enhance KG querying: it translates natural language into SPARQL queries, retrieves the corresponding triples, and grounds the LLM's answers in verified data.

Another promising architecture treats the LLM as an agent capable of tool use. In this configuration, the model not only generates text but can also issue *calls* to external tools or APIs – in our case, a graph query engine. The emerging Model Context Protocol (MCP) exemplifies this strategy. It defines a standardised method for an AI assistant to request data from external systems and incorporate it dynamically into its conversational context. Introduced by Anthropic in late 2024, MCP is described as ‘*a new standard for connecting AI assistants to the systems where data lives, including content repositories, business tools, and development environments*’ (Anthropic 2024). In essence, MCP allows an LLM – such as Claude (Anthropic 2025) – to query a database or a KG by specifying exactly what it needs and to receive structured results in real time during its reasoning process. By providing ‘*a universal, open standard for connecting AI systems with data sources*’, MCP aims to replace fragmented, ad hoc integrations with a consistent, interoperable interface. For example, under the MCP framework, an AI could invoke a ‘Neo4j query’ tool by submitting a Cypher query as a payload; the MCP server would then execute the query on the graph database and return the results to the AI, allowing it to continue the conversation with those results in memory (Figure 135).

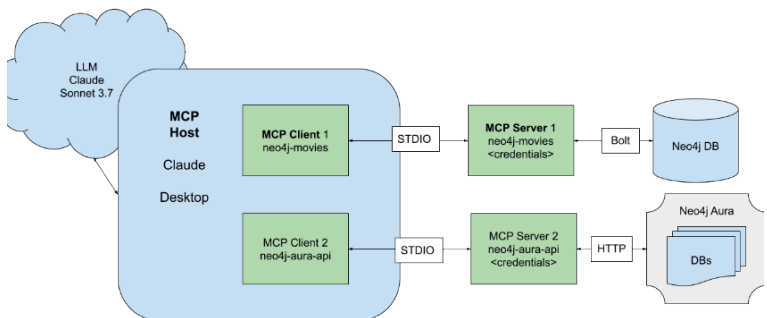
Beyond RAG and MCP-based agent architectures, other patterns include fine-tuning or pre-training LLMs on graph-structured data. In principle, it is possible to serialise parts of a KG into text and use them as base data to help the model absorb domain-specific knowledge. However, this approach may be static – the model's internalised knowledge may quickly become outdated as data evolves – and lacks transparency, as it is difficult to determine which facts the model actually retains. For these reasons, most current research favours *dynamic integration*: keeping authoritative data within the KG and letting the

LLM query it when needed. This *dynamic* approach also aligns with the federated ontology strategy introduced earlier in the book. Rather than embedding a monolithic schema within an AI's parameters, multiple domain ontologies are kept modular yet semantically connected in a live graph. This structure maintains flexibility and updatability and reflects the multi-scale and cross-disciplinary nature of AECO knowledge itself.

A further key architectural consideration concerns the direction of information flow. The system may rely on a one-way model, in which the KG informs the LLM through retrieval mechanisms. Alternatively, the flow can be bidirectional, allowing the LLM's outputs to be written back into and thus update the KG. In an interactive design assistant, for instance, the AI might propose a design solution that then needs to be evaluated, validated, or stored within the project's KG for future reference. This kind of *bidirectional integration* remains an area of active research. The MCP model already supports two-way connections – AI tools can both read and write via defined operations – hinting at future scenarios in which AI agents not only consult KG but also help to populate or update it.

In any case, the common goal of these architectures remains the same: to ensure that the LLM remains *grounded* in the KG's knowledge at all times while still offering fluid, natural, and context-aware conversational abilities to the end user. We now turn to a concrete example that illustrates how these concepts can take shape in practice.

Figure 135. Model Context Protocol Architecture for integrating Claude (LLM) with Neo4j (graph DB). Source: (Neo4j 2025)



10.6 Demonstration: Claude, MCP, and Neo4j

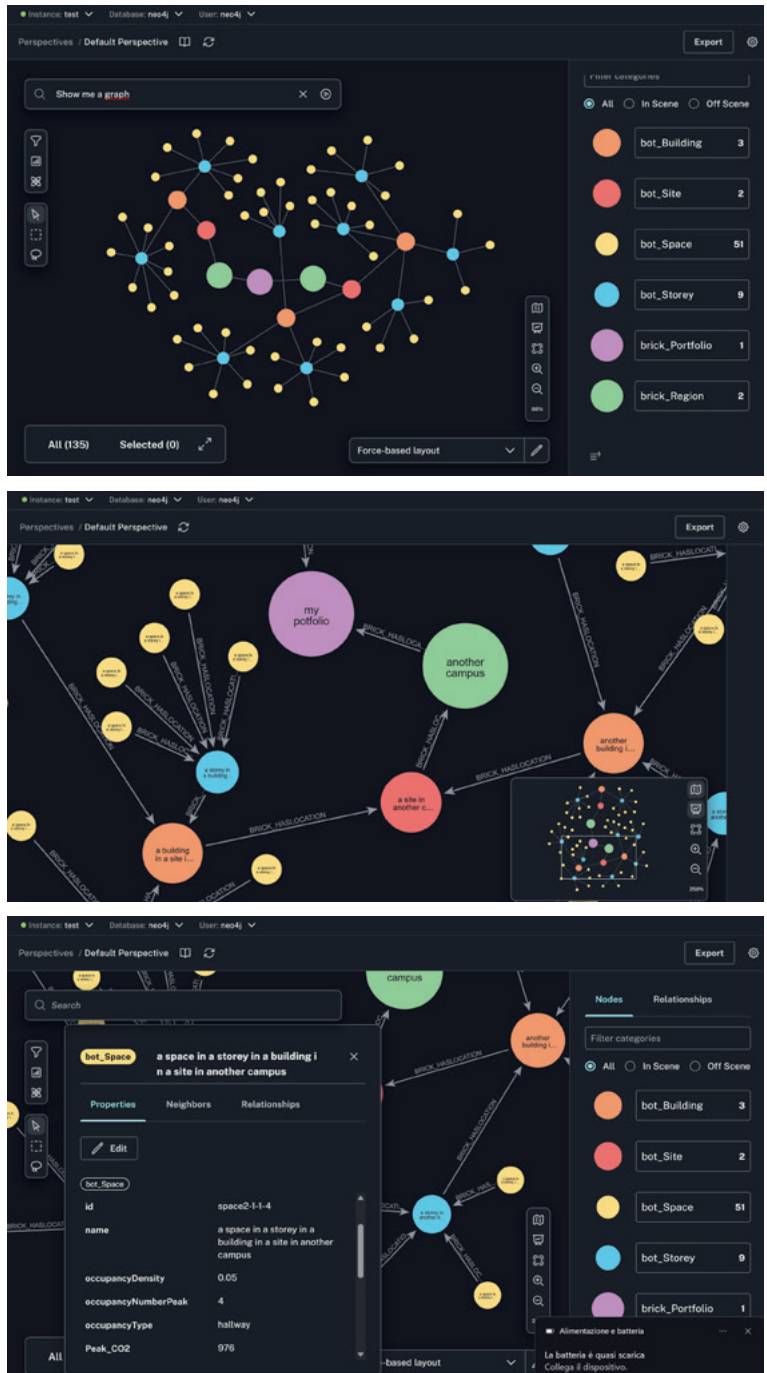
To demonstrate how an AI-integrated knowledge system might operate in practice, we will consider a fictional yet realistic scenario built upon the tools and concepts discussed in earlier chapters.

Imagine a small university building portfolio undergoing a performance evaluation for an energy retrofit. The portfolio consists of two campuses, each containing one or more sites. The first campus includes a single building with two storeys. The second campus has two buildings, with three and four storeys respectively. Each building houses spaces with various functions: classrooms, offices, toilets, corridors, hallways, laboratories, etc.

The facility managers have been collecting data from various systems. A detailed BIM model (in IFC format) provides the building's spatial and construction information. Sensors and meters record operational data such as temperature, humidity and CO₂ concentration. Building performance simulation software has been used to model different retrofit scenarios and compute heating performance indicators.

Using the BTwin toolkit, these heterogeneous data sources were integrated into a comprehensive Neo4j graph DB (Figure 136). The graph is built upon the federated ontology, ensuring that all data shares a consistent semantic context. In practice, this means that spatial elements, their properties, and associated performance indicators are represented as interconnected nodes in Neo4j. For instance, nodes such as `bot:Site`, `bot:Building`, and `bot:Space` define the spatial hierarchy; `ifc:PropertySet` nodes describe space properties; and `eko:KPI` nodes capture KPIs such as annual heating energy or average CO₂ concentration. Each ontology module contributes its respective part: BOT for geometry, Brick for sensor classification, IFC for construction detail, and EKO for performance metrics, etc., all unified through the federated model.

Figure 136. Neo4j graph for the fictional building portfolio.



Data is ingested into the graph via JSON-LD-formatted files. The BTwin connectors extract entities such as `ifcBuildingStorey` and `ifcSpace` from the IFC model, then link them to sensor data and simulation results via Python routines. As a result, the Neo4j database now serves as a core digital twin for the building portfolio's operational performance. For example, one portion of the graph contains nodes representing thermal zones defined in the EnergyPlus simulation, each linked to its corresponding physical space and associated with properties such as annual heating demand. The EKO ontology's `eko:hasAssociatedObject` relationship connects these performance metric nodes to the spaces or zones they describe. Temporal data is managed through `time:Interval` nodes, which represent simulation periods or sensor timestamps. In short, the KG is ready to answer complex, semantically grounded queries.

Now, the user enters Claude, an advanced LLM integrated into the graph system via the MCP. MCP enables direct connection between Claude's chat interface and the Neo4j database through the `neo4-mcp-server` (Neo4j 2025b). Claude is provided with a list of available tools – notably, *read* and *write* query tools – allowing it to issue Cypher queries to Neo4j and retrieve structured results.

The user interacts with Claude through natural language; Claude then interprets the request and translates it into a Cypher query consistent with Neo4j's schema and the federated ontology's vocabulary.

Let's consider three example queries, each targeting a different layer of the data:

- **Q1 – User:** «How many classrooms are in the portfolio? How many people can they host in total?»
- **Q2 – User:** «What is the highest peak CO₂ level among the classrooms, and which classroom recorded it? Can you show with a chart the classrooms over 1000ppm?»
- **Q3 – User:** «Can you identify the top three zones with the highest heating energy use? Please make a summary table and display with a chart.»

The results are shown in the following figures.

Figure 137. Claude's answer to Q1: «How many classrooms are in the portfolio? How many people can they host in total?».

School facility portfolio overview Condividi

A How many classrooms are in the portfolio? How many people can they host in total?

I'll query the Neo4j database to find information about classrooms in the portfolio.

N Get Neo4j Schema ⚠

Let me try a different approach to explore the database structure and find classroom information.

N Read Neo4j Cypher ▼

N Read Neo4j Cypher ▼

Great! Now I can see that spaces have an `occupancyType` property that includes "classroom" and an `occupancyNumberPeak` property that shows the maximum occupancy. Let me query for all classrooms.


N Read Neo4j Cypher ▼

Based on the Neo4j database query, the portfolio contains:

- **6 classrooms**
- **Total capacity: 285 people** (combined peak occupancy)

These classrooms have a peak occupancy capacity ranging across the portfolio, which represents the maximum number of people that can be hosted in all classrooms combined.

📄 👍 🗨 Riprova ▼

 Claude può commettere errori. Verifica sempre le risposte con attenzione.

Rispondi a Claude...

+ ⌵ 🕒 Sonnet 4.5 ▼ ⬆

School facility portfolio overview Condividi

A How many classrooms are in the portfolio? How many people can they host in total?

I'll query the Neo4j database to find information about classrooms in the portfolio.

N Get Neo4j Schema ⚠

Let me try a different approach to explore the database structure and find classroom information.

N Read Neo4j Cypher ▼

N Read Neo4j Cypher ▼

Great! Now I can see that spaces have an `occupancyType` property that includes "classroom" and an `occupancyNumberPeak` property that shows the maximum occupancy. Let me query for all classrooms.


N Read Neo4j Cypher ▼

Based on the Neo4j database query, the portfolio contains:

- **6 classrooms**
- **Total capacity: 285 people** (combined peak occupancy)

These classrooms have a peak occupancy capacity ranging across the portfolio, which represents the maximum number of people that can be hosted in all classrooms combined.

📄 👍 🗨 Riprova ▼

 Claude può commettere errori. Verifica sempre le risposte con attenzione.

Rispondi a Claude...

+ ⚙ 🕒 Sonnet 4.5 ▼ ⬆

Figure 138. Claude's answer to Q2: «What is the highest peak CO2 level among the classrooms, and which classroom recorded it? Can you show with a chart the classrooms over 1000ppm?».

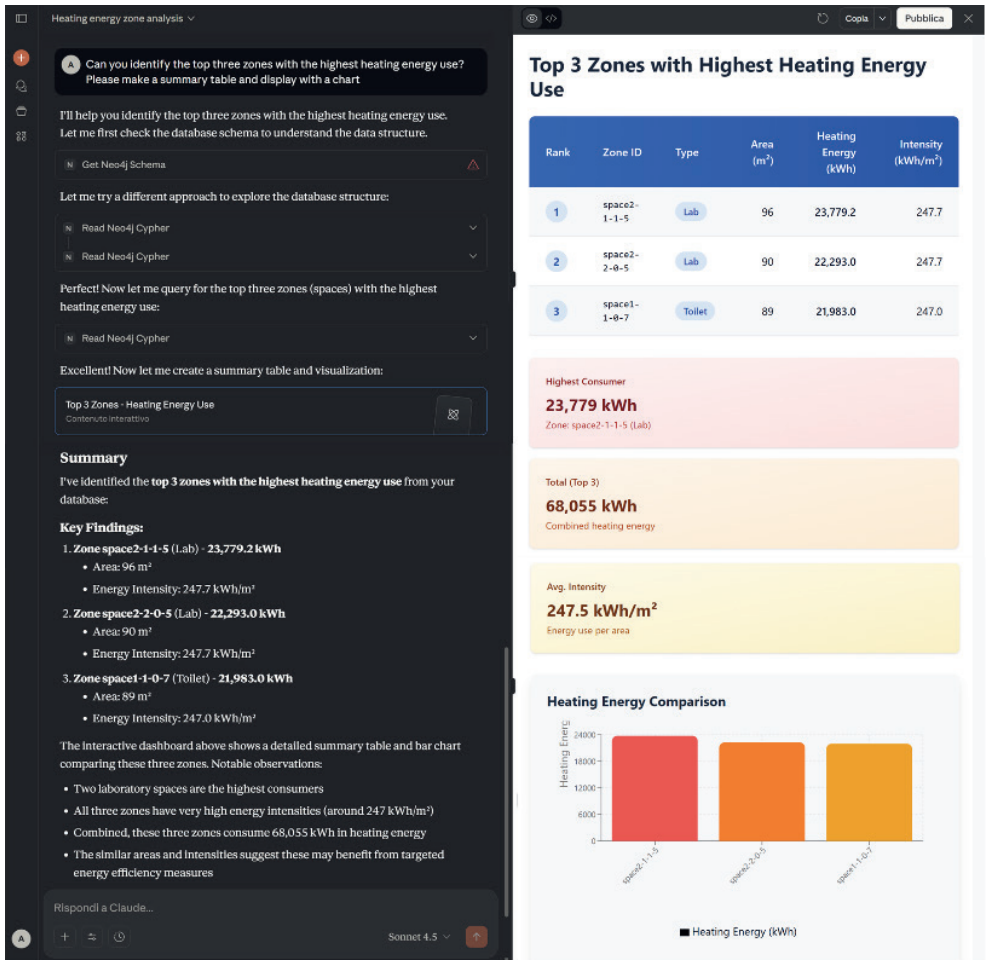


Figure 139. Claude's answer to Q3. «Can you identify the top three zones with the highest heating energy use? Please make a summary table and display with a chart».

When Claude receives these queries, several steps occur under the hood. First, it interprets the user's intent. For Q3, for instance, the user seeks the zones in <Building A> with the highest heating energy consumption and their respective occupancy levels – perhaps to assess energy efficiency per occupant. Claude identifies the relevant data structures through the ontology and decides to use the *read query tool* to fetch the required information. Through MCP, Claude issues a query to Neo4j. Thanks to the semantic vocabulary defined in the federated ontology, Claude does not need to infer the data structure: it already knows that heating energy is represented as an `eko:KPI` linked to spaces, and occupancy as another KPI or property.

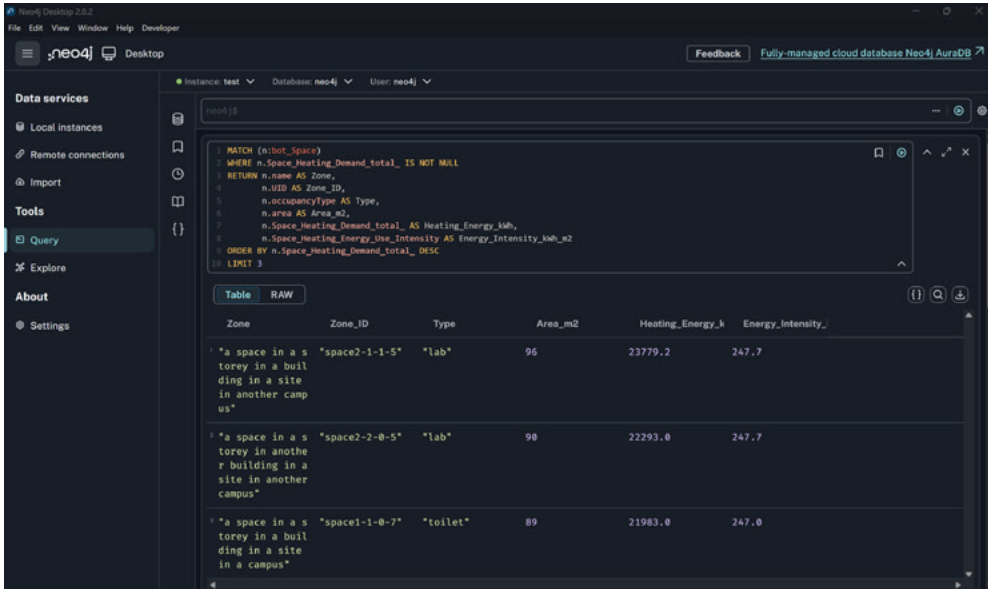
It therefore constructs a Cypher query such as the following (Figure 140 and Figure 141).

This query matches *Zone* nodes in *Building A*, finds their associated heating energy and occupancy KPI values, and returns the top three by heating energy demand.

The Neo4j MCP server executes the query and returns the following results (Figure 142).

```
MATCH (n:bot_Space)
WHERE n.Space_Heating_Demand_total_ IS NOT NULL
RETURN n.name AS Zone,
       n.UID AS Zone_ID,
       n.occupancyType AS Type,
       n.area AS Area_m2,
       n.Space_Heating_Demand_total_ AS Heating_Energy_kWh,
       n.Space_Heating_Energy_Use_Intensity AS Energy_Intensity_kWh_m2
ORDER BY n.Space_Heating_Demand_total_ DESC
LIMIT 3
```

Figure 140. Cypher query performed by Claude to get the three zones with the highest energy use in the portfolio.



The screenshot shows the Neo4j Desktop interface. The top menu includes File, Edit, View, Window, Help, and Developer. The main window displays a Cypher query in the editor and its results in a table view. The query is as follows:

```

MATCH (n:bot_Space)
WHERE n.Space_Heating_Demand_total_IS NOT NULL
RETURN n.name AS Zone,
       n.UUID AS Zone_ID,
       n.occupancyType AS Type,
       n.area AS Area_m2,
       n.Space_Heating_Demand_total AS Heating_Energy_kWh,
       n.Space_Heating_Energy_Use_Intensity AS Energy_Intensity_kWh_m2
ORDER BY n.Space_Heating_Demand_total_DESC
LIMIT 3
  
```

The results are displayed in a table with the following columns: Zone, Zone_ID, Type, Area_m2, Heating_Energy_k, and Energy_Intensity_.

Zone	Zone_ID	Type	Area_m2	Heating_Energy_k	Energy_Intensity_
"a space in a storey in a building in a site in another campus"	"space2-1-1-5"	"lab"	96	23779.2	247.7
"a space in a storey in another building in a site in another campus"	"space2-2-0-5"	"lab"	90	22293.0	247.7
"a space in a storey in a building in a site in a campus"	"space1-1-0-7"	"toilet"	89	21983.0	247.0

Figure 141. The same Cypher query applied in Neo4j.

Figure 142. Results from the Cypher query showing the three zones with the highest consumption in the portfolio.

```

[
  {
    "Zone_ID": "space2-1-1-5",
    "Type": "lab",
    "Area_m2": 96,
    "Heating_Energy_kWh": 23779.2,
    "Energy_Intensity_kWh_m2": 247.7
  },
  {
    "Zone_ID": "space2-2-0-5",
    "Type": "lab",
    "Area_m2": 90,
    "Heating_Energy_kWh": 22293.0,
    "Energy_Intensity_kWh_m2": 247.7
  },
  {
    "Zone_ID": "space1-1-0-7",
    "Type": "classroom",
    "Area_m2": 89,
    "Heating_Energy_kWh": 21983.0,
    "Energy_Intensity_kWh_m2": 247.0
  }
]
  
```

Then, Claude receives these results and composes a response. Importantly, it does not simply list the raw data: it uses its language generation capability to produce an informative and contextualised message:

A3 - Claude: *I've identified the top 3 zones with the highest heating energy use from your database:*

Key Findings:

1. *Zone space2-1-1-5 (Lab) - 23,779.2 kWh*
 - *Area: 96 m²*
 - *Energy Intensity: 247.7 kWh/m²*
2. *Zone space2-2-0-5 (Lab) - 22,293.0 kWh*
 - *Area: 90 m²*
 - *Energy Intensity: 247.7 kWh/m²*
3. *Zone space1-1-0-7 (Classroom) - 21,983.0 kWh*
 - *Area: 89 m²*
 - *Energy Intensity: 247.0 kWh/m².*

This exchange exemplifies human–AI collaboration. The AI navigates the KG autonomously and presents findings in a clear, reasoned manner. Note how the LLM adds plausible interpretation (suggesting causes for high per-person energy use) – something a raw database query could never provide. Yet every factual element in the response is grounded in the KG, ensuring traceability and validity. If desired, the user could even request evidence.

Let's consider now a follow-up query like:

- **Q4 - User:** *«What retrofit actions would you suggest for those zones to reduce heating demand?»*

This is an open-ended question that requires reasoning rather than retrieval. Claude cannot extract a direct answer from the KG because the graph contains only factual information, not design strategies. Claude's answer is provided in the following figure.

In composing this response, Claude explicitly combined data retrieved from the KG with general knowledge acquired from its training corpus. However, its reliability remains limited: since the graph lacks explicit data on the efficiency of construction systems or HVAC components, there is nothing to allow Claude

to determine which retrofit strategies are most appropriate for each space. In other words, it is hallucinating. A more appropriate reply would have been something along the lines of: «*Based on the data I have available, I cannot know.*» Thus, while the conversation demonstrates the potential of AI-KG integration, it also exposes the limitations of current implementations, where an LLM's inferential abilities still depend on the depth and completeness of the underlying knowledge base. At the same time, it underscores the importance of human awareness and critical judgment in evaluating AI-generated answers.

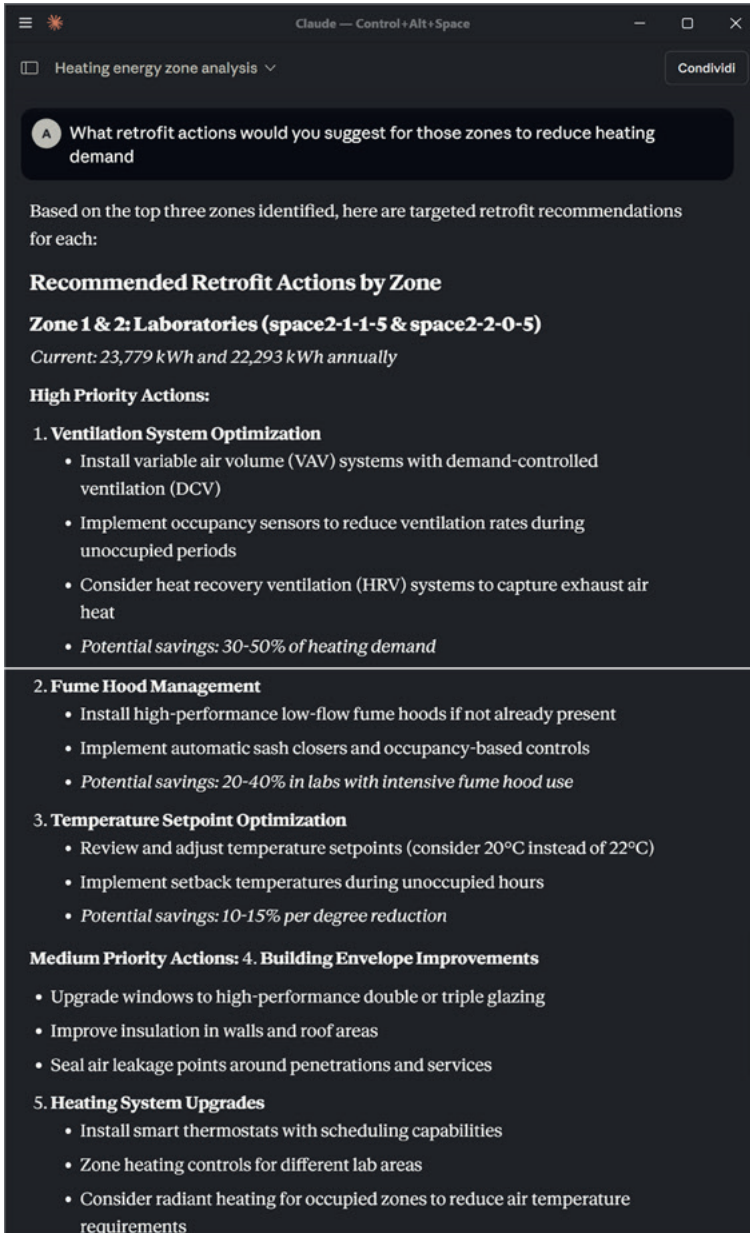
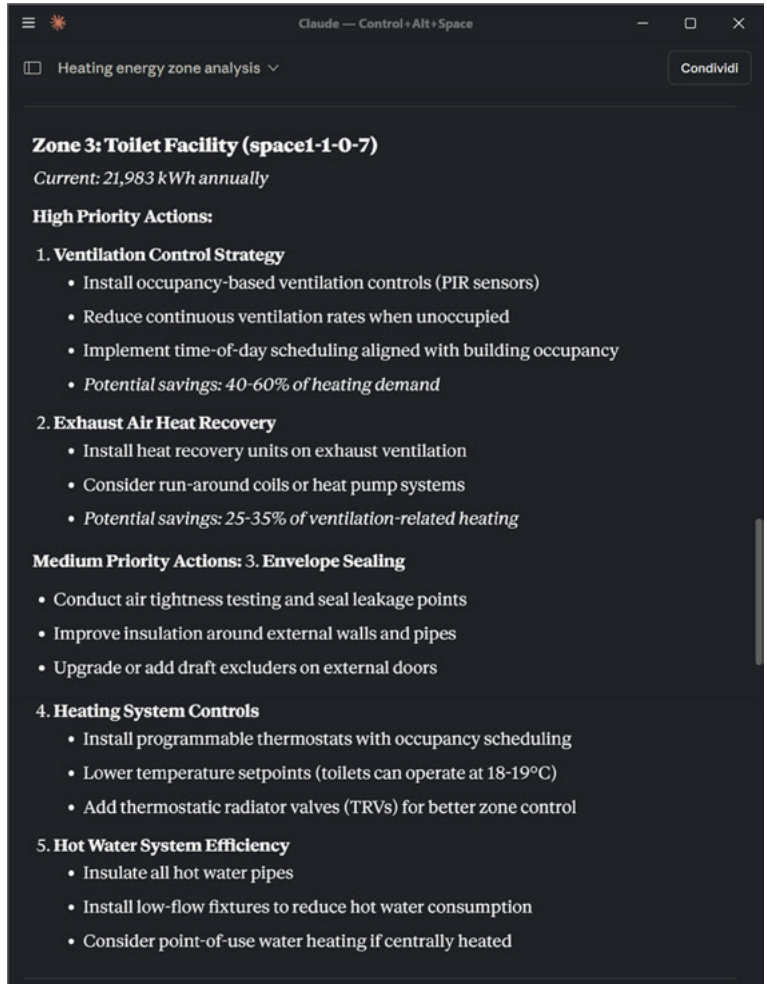


Figure 143. Claude's answer to Q4 (Part 1).

Figure 144. Claude's answer to Q4 (Part 2).



The screenshot shows a dark-themed window titled "Claude — Control+Alt+Space". The main content is a report titled "Heating energy zone analysis" with a "Condividi" button in the top right. The report is for "Zone 3: Toilet Facility (space1-1-0-7)" and states a current consumption of "21,983 kWh annually". It lists "High Priority Actions" in two numbered sections: 1. Ventilation Control Strategy and 2. Exhaust Air Heat Recovery. Below these are "Medium Priority Actions" in three numbered sections: 3. Envelope Sealing, 4. Heating System Controls, and 5. Hot Water System Efficiency. Each section contains a list of specific recommendations.

☐ Heating energy zone analysis Condividi

Zone 3: Toilet Facility (space1-1-0-7)
Current: 21,983 kWh annually

High Priority Actions:

- 1. Ventilation Control Strategy**
 - Install occupancy-based ventilation controls (PIR sensors)
 - Reduce continuous ventilation rates when unoccupied
 - Implement time-of-day scheduling aligned with building occupancy
 - *Potential savings: 40-60% of heating demand*
- 2. Exhaust Air Heat Recovery**
 - Install heat recovery units on exhaust ventilation
 - Consider run-around coils or heat pump systems
 - *Potential savings: 25-35% of ventilation-related heating*

Medium Priority Actions:

- 3. Envelope Sealing**
 - Conduct air tightness testing and seal leakage points
 - Improve insulation around external walls and pipes
 - Upgrade or add draft excluders on external doors
- 4. Heating System Controls**
 - Install programmable thermostats with occupancy scheduling
 - Lower temperature setpoints (toilets can operate at 18-19°C)
 - Add thermostatic radiator valves (TRVs) for better zone control
- 5. Hot Water System Efficiency**
 - Insulate all hot water pipes
 - Install low-flow fixtures to reduce hot water consumption
 - Consider point-of-use water heating if centrally heated

Claude — Control+Alt+Space

☐ Heating energy zone analysis ▾ Condividi

Cross-Cutting Recommendations for All Zones

- Building Management System (BMS) Integration**
 - Centralized monitoring and control of all three zones
 - Real-time energy monitoring and anomaly detection
 - Automated optimization based on occupancy patterns
- Commissioning and Maintenance**
 - Conduct comprehensive HVAC system commissioning
 - Regular maintenance schedules to ensure optimal performance
 - Staff training on efficient system operation
- Renewable Heating Integration**
 - Explore air-source or ground-source heat pumps
 - Solar thermal pre-heating for ventilation air
 - Consider district heating connections if available

Expected Overall Impact

Zone	Current Use (kWh/yr)	Estimated Savings	Projected Use (kWh/yr)
Lab space2-1-1-5	23,779	30-40%	14,267-16,645
Lab space2-2-0-5	22,293	30-40%	13,376-15,605
Toilet space1-1-0-7	21,983	35-45%	12,091-14,289
Total	68,055	32-42%	39,734-46,539

Priority Implementation Order:

- Start with ventilation controls (lowest cost, highest impact)
- Proceed with heat recovery systems
- Implement BMS integration for ongoing optimization
- Consider envelope improvements during scheduled renovations

Would you like me to provide more detailed specifications for any of these retrofit measures or help you prioritize based on budget constraints?

☰ ☺ ☑ Riprova ▾

Claude può commettere errori. Verifica sempre le risposte con attenzione.

Rispondi a Claude...

+ ☰ ⌚ Sonnet 4.5 ▾ ↗

Figure 145. Claude's answer to Q4 (Part 3).

10.7 Future scenarios of AI-integrated knowledge in AECO

Looking ahead, it is entirely plausible to imagine AI-integrated knowledge systems becoming a ubiquitous component of the digital ecosystem within the built environment. A near-term scenario in building management might involve integrating LLM-powered assistants into digital twin platforms (Luo *et al.* 2024) a system that employs Large Language Models (LLMs). As digital twins continuously monitor building performance, LLMs could serve as intelligent mediators, allowing stakeholders to interact with the system using natural language.

Imagine, for example, a city's building operations centre where a manager asks: «*Which buildings in our portfolio are likely to experience comfort issues next week based on the weather forecast?*» The AI assistant could cross-analyse the KGs of multiple digital twins and return a ranked list with explanatory notes. A federated ontology approach would ensure interoperability across diverse systems, enabling the comparison of key metrics such as temperature, humidity, and logged complaints, even when data structures vary.

While awaiting the commercial availability of such technologies, the focus should therefore be on federated ontology and knowledge modelling. These will be essential prerequisites, serving as the semantic backbone for such scenarios and enabling cross-domain queries that, at present, would require specialist teams to coordinate. Future developments will also need to include mechanisms for checking the validity of AI outputs – undoubtedly a fundamental requirement for moving AI from experimental prototypes to robust, real-world products. One approach could be to encode rules within the KG, allowing AI agents to 'self-censor' or verify their responses. Also of significant interest are multi-agent systems, which are systems orchestrated by a lead agent working with various specialised delegates: one executing the query, another validating the plausibility of the results, another exporting the data in a required format, and so forth. Likely, we will also see the development of specialised benchmark datasets for building-specific question answering, such as question sets derived from building codes, with correct answers based on authoritative sources. These would help push LLMs towards greater factual accuracy in our

domain. We will also become accustomed to the idea that software interfaces will no longer be static. As LLMs are already capable of generating websites, dashboards and applications on the fly (see Claude, for example), it is likely that each user will have a customised data interface tailored to their real-time requests via chat. In practice, future AI-integrated systems will likely operate in a multimodal and adaptive fashion.

Beyond the technical explorations and future visions, one crucial constant remains: human expertise will remain central. Technical tasks can increasingly be delegated to AI, yet responsibility for interpretation and critical analysis cannot be outsourced. AI-integrated knowledge systems will undoubtedly manage information retrieval and pattern recognition on a scale far beyond human capability. Awareness will therefore probably be the true value that an architect or engineer contributes. Moreover, determining which questions to ask through the prompts, which recommendations to follow, how to implement solutions, and, more broadly, whether and how to act on such outputs will hopefully still rely on human judgement, ethics, and creativity.

It is not difficult to hypothesise that in a future scenario, the role of the AECO professional may shift from painstaking data gathering to the curation and validation of AI-generated insights. This has major implications for education and research: future professionals should be trained not only in their core disciplines but also in how to work with AI – understanding its limitations, verifying its outputs, and contributing to the development of the underlying knowledge models. In essence, they should learn how to use these tools responsibly and consciously.

None of this will happen, however, unless collaboration and standardisation across the industry progress at sufficient speed. If AI-integrated systems proliferate too quickly and without coordination, the resulting complexity may become unmanageable. It is therefore essential, now more than ever, to support initiatives aimed at creating a shared discipline for digital knowledge modelling in the built environment. These efforts will enable even smaller firms and public sector organisations to understand and adopt such technologies without having to start from scratch. It is conceivable that in less than a decade, asking an AI assistant about a building will be as routine as

checking one's email. But behind that apparent simplicity will probably lie a sophisticated, domain-structured network of data, information, standards, models, and validation protocols, a disciplinary infrastructure that the AECO community will be continuously called to design, build, maintain, and evolve actively.

REFERENCES

ABUIMARA ET AL., 2021

Abuimara T, Hobson BW, Gunay B, O'Brien W and Kane M (2021) Current state and future challenges in building management: Practitioner interviews and a literature review. *Journal of Building Engineering* 41, 102803. <https://doi.org/10.1016/j.jobe.2021.102803>.

AL-ADDOUS AND ALBATAYNEH, 2020

Al-Addous M and Albatayneh A (2020) Knowledge gap with the existing building energy assessment systems. *Energy Exploration & Exploitation* 38(3), 783–794. <https://doi.org/10.1177/0144598719888100>.

ALVES, PALHA AND ALMEIDA, 2025

Alves JL, Palha RP and Almeida Filho ATD (2025) Towards an integrative framework for BIM and artificial intelligence capabilities in smart architecture, engineering, construction, and operations projects. *Automation in Construction* 174, 106168. <https://doi.org/10.1016/j.autcon.2025.106168>.

ANTHROPIC, 2024

Anthropic (2024, November) Introducing the Model Context Protocol (MCP). <https://www.anthropic.com/news/model-context-protocol>

ANTHROPIC, 2025

Anthropic (2025) Claude AI. <https://claude.ai/>

ANTUNES ET AL., 2024

Antunes MLR, César KML, Ribeiro JCL, Oliveira DSD and Carvalho JMFD (2024) Analysis of IFC interoperability data schema for project representation. *Automation in Construction* 166, 105650. <https://doi.org/10.1016/j.autcon.2024.105650>.

ARAZZI ET AL., 2025

Arazzi M, Ligari D, Nicolazzo S and Nocera A (2025) Augmented Knowledge Graph Querying leveraging LLMs. (Version 1). arXiv. <https://doi.org/10.48550/ARXIV.2502.01298>.

ARGHAVAN ET AL., 2025

Arghavan AA, Osadcha I, Farghaly K, Bosché F and Puust R (2025) Development of a Built Environment Ontology Lookup Service (BE-OLS) with a New Ontology Evaluation. Presented at the 2025 European Conference on Computing in Construction. <https://doi.org/10.35490/EC3.2025.353>.

BADER, 2022

Bader DA (ed) (2022) *Massive graph analytics*, First edition. Place of publication not identified: Chapman and Hall/CRC.

BALAJI ET AL., 2016

Balaji B, Bhattacharya A, Fierro G, Gao J, Gluck J, Hong D, Johansen A, Koh J, Ploennigs J, Agarwal Y, Berges M, Culler D, Gupta R, Kjærsgaard MB, Srivastava M and Whitehouse K (2016) Brick: Towards a Unified Metadata Schema For Buildings. In *Proceedings of the 3rd ACM International Conference on Systems For Energy-Efficient Built Environments*. Palo Alto CA USA: ACM, 41–50. <https://doi.org/10.1145/2993422.2993577>.

BALAJI ET AL., 2018

Balaji B, Bhattacharya A, Fierro G, Gao J, Gluck J, Hong D, Johansen A, Koh J, Ploennigs J, Agarwal Y, Bergés M, Culler D, Gupta RK, Kjærsgaard MB, Srivastava M and Whitehouse K (2018) Brick : Metadata schema for portable smart building applications. *Applied Energy* 226, 1273–1292. <https://doi.org/10.1016/j.apenergy.2018.02.091>.

BARABÁSI AND PÓSFAL, 2016

Barabási A-L and Pósfai M (2016) *Network science*. Cambridge: Cambridge university press.

BELLINI, NESI AND SODERI, 2018

Bellini P, Nesi P and Soderi M (2018) Km4City Ontology – Knowledge Model 4 the City (v1.6.x). DISIT Lab, University of Florence. <https://www.disit.org/km4city/schema>

BENEDIKT, 1979

Benedikt ML (1979) To Take Hold of Space: Isovists and Isovist Fields. *Environment and Planning B* 6(1), 47–65. <https://doi.org/10.1068/b060047>.

BENFER AND MÜLLER, 2024

Benfer R and Müller J (2024) Semantic digital twin creation of building systems through time series based metadata inference – A review. *Energy and Buildings* 321, 114637. <https://doi.org/10.1016/j.enbuild.2024.114637>.

BERKOBEN ET AL., 2020

Berkoben K, Kaed C, Sodorff T and Team GDB (2020) Digital Buildings Ontology (DBO) – a semantic data model for managing building-installed equipment and spaces. <https://google.github.io/digitalbuildings/ontology/>

BETTI ET AL., 2024

Betti G, Tartarini F, Nguyen C and Schiavon S (2024) CBE Clima Tool: A free and open-source web application for climate analysis tailored to sustainable building design. *Building Simulation* 17(3), 493–508. <https://doi.org/10.1007/s12273-023-1090-5>.

BIM DICTIONARY, 2020

BIM Dictionary (2020) Digital Twin. <https://bimdictionary.com/terms/search>

BIZER, HEATH AND BERNERS-LEE, 2009

Bizer C, Heath T and Berners-Lee T (2009) Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems* 5(3), 1–22. <https://doi.org/10.4018/jswis.2009081901>.

BOJE ET AL., 2020

Boje C, Guerriero A, Kubicki S and Rezgui Y (2020) Towards a semantic Construction Digital Twin: Directions for future research. *Automation in Construction* 114, 103179. <https://doi.org/10.1016/j.autcon.2020.103179>.

BUILDINGSMART, 2025

buildingSMART International (2025) IFC 4.3.2.20251007 (IFC4X3_ADD2). <https://ifc43-docs.standards.buildingsmart.org/> (accessed 26 October 2025)

BUILDINGSMART, 2025

buildingSMART International (2025) IFC Formats. <https://technical.buildingsmart.org/standards/ifc/ifc-formats/> (accessed 29 October 2025)

CHAMARI, PETROVA AND PAUWELS, 2022

Chamari L, Petrova E and Pauwels P (2022) A web-based approach to BMS, BIM and IoT integration. In 2022: *CLIMA 2022 The 14th REHVA HVAC World Congress*.

CHAMARI, PETROVA AND PAUWELS, 2023

Chamari L, Petrova E and Pauwels P (2023) An End-to-End Implementation of a Service-Oriented Architecture for Data-Driven Smart Buildings. *IEEE Access* 11, 117261–117281. <https://doi.org/10.1109/ACCESS.2023.3325767>.

CHARPENAY, 2020

Charpenay V (2020) Haystack Tagging Ontology (HTO) — version 0.2.0. <https://www.vcharpenay.link/hto/doc.htm>

CHEN ET AL., 2019

Chen Y, Guan Z, Zhang R, Du X and Wang Y (2019) A survey on visualization approaches for exploring association relationships in graph data. *Journal of Visualization* 22(3), 625–639. <https://doi.org/10.1007/s12650-019-00551-y>.

CHENG AND SUN, 2025

Cheng K and Sun Y (2025) Complex Query Answering. In Cheng K and Sun Y, *Knowledge Graph Reasoning*. Cham: Springer Nature Switzerland, 73–106. https://doi.org/10.1007/978-3-031-72008-6_4.

COMMUNITY, 2025

Community PH (2025) Project Haystack: An Open Source Initiative to Streamline Working with IoT Data. Project Haystack. <https://www.project-haystack.org/>

COMPTON ET AL., 2012

Compton M, Barnaghi P, Bermudez L, García-Castro R, Corcho O, Cox S, Graybeal J, Hauswirth M, Henson C, Herzog A, Huang V, Janowicz K, Kelsey WD, Le Phuoc D, Lefort L, Leggieri M, Neuhaus H, Nikolov A, Page K, Passant A, Sheth A and Taylor K (2012) The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics* 17, 25–32. <https://doi.org/10.1016/j.websem.2012.05.003>.

CRUCITTI, LATORA AND PORTA, 2006

Crucitti P, Latora V and Porta S (2006) Centrality measures in spatial networks of urban streets. *Physical Review E* 73(3), 036125. <https://doi.org/10.1103/PhysRevE.73.036125>.

D'AQUIN, 2025

d'Aquin M (2025) On the role of knowledge graphs in AI-based scientific discovery. *Journal of Web Semantics* 84, 100854. <https://doi.org/10.1016/j.websem.2024.100854>.

DAS AND SOYLU, 2023

Das R and Soylu M (2023) A key review on graph data science: The power of graphs in scientific studies. *Chemometrics and Intelligent Laboratory Systems* 240, 104896. <https://doi.org/10.1016/j.chemolab.2023.104896>.

DAWKINS, HUDSON-SMITH AND DENNET, 2018

Dawkins O, Hudson-Smith A and Dennet A (2018) Living with a Digital Twin: Operational management and engagement using IoT and Mixed Realities at UCL's Here East Campus on the Queen Elizabeth Olympic Park. In *Proceedings of the 26th annual GIScience Research UK conference: GISRUK 2018*. GIS Research UK (GISRUK) University of Leicester, UK.

DE WILDE, 2023

de Wilde P (2023) Building performance simulation in the brave new world of artificial intelligence and digital twins: A systematic review. *Energy and Buildings* 292, 113171. <https://doi.org/10.1016/j.enbuild.2023.113171>.

DEAN AND SCHREIBER, 2004

Dean M and Schreiber G (2004) OWL Web Ontology Language Reference - W3C Recommendation 10 February 2004. https://www.w3.org/TR/owl-ref/?utm_source=chatgpt.com#Syntax (accessed 20 September 2025)

DENG, MENASSA AND KAMAT, 2021

Deng M, Menassa CC and Kamat VR (2021) From BIM to digital twins: a systematic review of the evolution of intelligent building representations in the AEC-FM industry. *Journal of Information Technology in Construction* 26, 58–83. <https://doi.org/10.36680/j.itcon.2021.005>.

DI ET AL., 2023

Di Turi S, Palladino D, Caffari F, Centi G, Margiotta F, Ronchetti L, Signoretti P and Volpe L (2023) *Energy efficiency of complexes of historic and protected buildings: Characterisation of real case studies (LA1.4)*. ENEA – Italian National Agency for New Technologies, Energy and Sustainable Economic Development. (No. RdS_PTR22-24_PR1.5_LA1.4_073). Rome, Italy. Retrieved from <https://www.ricer-casistemaelettrico.enea.it/accordo-di-programma-mase-enea-2022-2024/de-carbonizzazione/edifici-ad-alta-efficienza-per-la-transizione-energetica.html>

DI ET AL., 2024

Di Turi S, Palladino D, Murano G, Centi G, Ronchetti L and Signoretti P (2024) *Energy efficiency of complexes of historic and protected buildings: Energy audit of case studies and assessment of the retrofit interventions (LA1.5)*. ENEA – Italian National Agency for New Technologies, Energy and Sustainable Economic Development. (No. RdS_PTR22-24_PR1.5_LA1.5_207). Rome, Italy.

DORAN, 1981

Doran GT (1981) There's a SMART Way to Write Management's Goals and Objectives. *Journal of Management Review* 70, 35–36.

DU AND LUO, 2023

Du J and Luo T (2023) Digital Twin Graph: Automated Domain-Agnostic Construction, Fusion, and Simulation of IoT-Enabled World. (Version 1). arXiv. <https://doi.org/10.48550/ARXIV.2304.10018>.

EC3, 2025

EC3 M&S Committee 2025 (2025) Built Environment Ontology Lookup Service. <https://cyberbuildlab.github.io/BE-OLS/ontologyCards.html> (accessed 22 September 2025)

ESNAOLA-GONZALEZ AND DÍEZ, 2020

Esnaola-Gonzalez I and Díez FJ (2020) RESPOND Ontology – Version 1.0 (Release 2020-04-14). <https://respond-project.github.io/RESPOND-Ontology/respond/index-en.html>

EUN ET AL., 2024

Eun Jung D, Yoo S, Ho Lee K and Kim J (2024) In-situ measurement of thermal transmittance of building walls: Evaluation of stored heat flux in heavy-weight walls during the cooling season. *Energy and Buildings* 325, 114981. <https://doi.org/10.1016/j.enbuild.2024.114981>.

EVERYTHING, 2017

EVERYTHING (2017) Web Thing Model. <https://model.webofthings.io/> (accessed 26 October 2025)

FERIA AND VILLALÓN, N.D.

Feria SC and Villalón MV (n.d.) Key Performance Indicator ontology. <https://bimerr.iot.linkeddata.es/def/key-performance-indicator/> (accessed 8 August 2025)

CORPO NAZIONALE DEI VIGILI DEL FUOCO, 2022

Corpo Nazionale dei Vigili del Fuoco (2022) Codice di Prevenzione Incendi – Testo coordinato (aggiornato al 27 dicembre 2022). <https://www.vigilfuoco.it/servizi-le-aziende-e-i-professionisti/prevenzione-incendi/testi-coordinati-di-prevenzione/codice-prevenzione-incendi>

GALIMBERTI, 2023

Galimberti U (2023) *L'etica del viandante*. Feltrinelli.

GANGULI AND ADHIKARI, 2020

Ganguli R and Adhikari S (2020) The digital twin of discrete dynamic systems: Initial approaches and future challenges. *Applied Mathematical Modelling* 77, 1110–1128. <https://doi.org/10.1016/j.apm.2019.09.036>.

GERBER, NGUYEN AND GAETANI, 2019

Gerber D, Nguyen B and Gaetani I (2019) Digital Twin. Towards a Meaningful Framework. www.arup.com/digitaltwinreport

GHOSH ET AL., 2023

Ghosh S, Mallick A, Chowdhury A and De Sarkar K (2023) Graph Theory Applications in Advanced Geospatial Research. (Version 2). arXiv. <https://doi.org/10.48550/ARXIV.2309.03249>.

GOLBECK, 2013

Golbeck J (2013) Nodes, Edges, and Network Measures. In *Analyzing the Social Web*. Elsevier, 9–23. <https://doi.org/10.1016/B978-0-12-405531-5.00002-X>.

GRASSHOPPER, 2025

Grasshopper - algorithmic modeling for Rhino (2025). <https://www.grasshopper3d.com/> (accessed 6 November 2025)

GREEN AND INC., N.D.

Green Building XML (gbXML) Schema, Inc. (n.d.) gbXML Green Building - Current Version of the gbXML Schema. https://www.gbxml.org/Schema_Current_GreenBuildingXML_gbXML (accessed 26 October 2025)

GRIEVES AND VICKERS, 2017

Grieves M and Vickers J (2017) Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In Kahlen F-J, Flumerfelt S, and Alves A (eds), *Transdisciplinary Perspectives on Complex Systems*. Cham: Springer International Publishing, 85–113. https://doi.org/10.1007/978-3-319-38756-7_4.

GRUBER, 1993

Gruber TR (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(2), 199–220. <https://doi.org/10.1006/knac.1993.1008>.

GUI, GOU AND LU, 2021

Gui X, Gou Z and Lu Y (2021) Reducing university energy use beyond energy retrofiting: The academic calendar impacts. *Energy and Buildings* 231, 110647. <https://doi.org/10.1016/j.enbuild.2020.110647>.

GULLI AND PREDARI, 2018

Gulli R and Predari G (2018) Il Moderno a Bologna: la Facoltà di Ingegneria di Giuseppe Vaccaro (1932-1935). *ANANKE* 83, 101–108.

HAMMAR ET AL., 2019

Hammar K, Wallin EO, Karlberg P and Hälleberg D (2019) RealEstateCore (REC) — An open-source ontology for the built environment. RealEstateCore Consortium. <https://www.realestatecore.io/>

HOSAMO AND MAZZETTO, 2024

Hosamo H and Mazzetto S (2024) Integrating Knowledge Graphs and Digital Twins for Heritage Building Conservation. *Buildings* 15(1), 16. <https://doi.org/10.3390/buildings15010016>.

IFCOPENSHELL, N.D.

IfcOpenShell (n.d.) IfcOpenShell - The open source IFC toolkit and geometry engine. <https://ifcopenshell.org/> (accessed 2 November 2025)

ITALIAN, 2011

Italian National Institute of Statistics (ISTAT) (2011) Population and Housing Census. <http://dati-censimentopolazione.istat.it/Index.aspx>

JABI, 2024

Jabi W (2024, June 10) topologicpy. (Version 0.7.17). pypi.org. <https://doi.org/10.5281/ZENODO.11555172>.

JABI, 2025

Jabi W (2025) Modified Swiss Dwellings 01 JSON. A JSON derivation of the Modified Swiss Dwellings Dataset for TopologicPy. <https://www.kaggle.com/datasets/wassimjabi/modified-swiss-dwellings-01-json> (accessed 24 October 2025)

JABI AND ALYMANI, 2020

Jabi W and Alymani A (2020) Graph Machine Learning using 3D Topological Models. Presented at the SimAUD.

JABI AND CHATZIVASILEIADI, 2021

Jabi W and Chatzivasileiadi A (2021) Topologic: Exploring Spatial Reasoning Through Geometry, Topology, and Semantics. In Eloy S, Leite Viana D, Morais E, and Vieira Vaz J (eds), *Formal Methods in Architecture*. Cham: Springer International Publishing, 277–285. https://doi.org/10.1007/978-3-030-57509-0_25.

JANOWICZ ET AL., 2018

Janowicz K, Haller A, Cox SJD, Phuoc DL and Lefrancois M (2018) SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators. (Version 2). <https://doi.org/10.48550/ARXIV.1805.09979>.

KAU ET AL., 2024

Kau A, He X, Nambissan A, Astudillo A, Yin H and Aryani A (2024) Combining Knowledge Graphs and Large Language Models. (Version 1). arXiv. <https://doi.org/10.48550/ARXIV.2407.06564>.

KOEHOORN ET AL., 2017

Koehoorn J, Feng C, Kustra J, Jalba A and Telea A (2017) Unified part-patch segmentation of mesh shapes using surface skeletons. In *Skeletonization: Theory, Methods and Applications*. Amsterdam, Netherlands: Elsevier, 89–122. <https://doi.org/10.1016/B978-0-08-101291-8.00005-5>.

KUKKONEN ET AL., 2022

Kukkonen V, Küçükavcı A, Seidenschnur M, Rasmussen MH, Smith KM and Hviid CA (2022) An ontology to support flow system descriptions from design to operation of buildings. *Automation in Construction* 134, 104067. <https://doi.org/10.1016/j.autcon.2021.104067>.

LADYBUG, 2025

Ladybug Tools (2025) Welcome to Honeybee's documentation! <https://www.ladybug.tools/honeybee-core/docs/> (accessed 2 December 2025)

LEE, JANG AND HYUN, 2024

Lee G, Jang S and Hyun S (2024) A Generalized LLM-Augmented BIM Framework: Application to a Speech-to-BIM system. (Version 1). arXiv. <https://doi.org/10.48550/ARXIV.2409.18345>.

LEFRANÇOIS ET AL., 2024

Lefrançois M, Garcia-Castro R, Poveda-Villalón M, Bouter C, Biagioni G, Daniele L and Gnabasiq D (2024) SAREF: the Smart Applications REFERENCE Ontology — Core (version 4.1.1). ETSI. <https://saref.etsi.org/core/v4.1.1/>

LEHTOLA ET AL., 2022

Lehtola VV, Koeva M, Elberink SO, Raposo P, Virtanen J-P, Vahdatikhaki F and Borsci S (2022) Digital twin of a city: Review of technology serving city needs. *International Journal of Applied Earth Observation and Geoinformation* 114, 102915. <https://doi.org/10.1016/j.jag.2022.102915>.

LI ET AL., 2019

Li Y, Garcia-Castro R, Mihindukulasooriya N, O'Donnell J and Vega-Sánchez S (2019) Enhancing energy management at district and building levels via an EM-KPI ontology. *Automation in Construction* 99, 152–167. <https://doi.org/10.1016/j.autcon.2018.12.010>.

LU ET AL., 2022

Lu Q, Xie X, Parlikad AK, Schooling J and Pitt M (2022) *Digital twins in the built environment: fundamentals, principles and applications*. London: ice Publishing.

LU ET AL., 2020

Lu Q, Xie X, Parlikad AK and Schooling JM (2020) Digital twin-enabled anomaly detection for built asset monitoring in operation and maintenance. *Automation in Construction* 118, 103277. <https://doi.org/10.1016/j.autcon.2020.103277>.

LUO, PARN AND BRILAKIS, 2024

Luo P, Parn E and Brilakis I (2024, July 14) ChatTwin: Enabling Natural Language Interactions with Infrastructure Digital Twins. European Council for Computing in Construction. <https://doi.org/10.17863/CAM.107986>.

LYGERAKIS, KAMPELIS AND KOLOKOTSA, 2022

Lygerakis F, Kampelis N and Kolokotsa D (2022) Knowledge Graphs' Ontologies and Applications for Energy Efficiency in Buildings: A Review. *Energies* 15(20), 7520. <https://doi.org/10.3390/en15207520>.

MASSACHUSETTS, FROM AND 2017., N.D.

Massachusetts Institute of Technology, From Plan to Action: MIT Campus Greenhouse Gas Emissions Reduction Strategy, 2017. (n.d.).

MASSAFRA ET AL., 2023

Massafra A, Costantino C, Predari G and Gulli R (2023) Building Information Modeling and Building Performance Simulation-Based Decision Support Systems for Improved Built Heritage Operation. *Sustainability* 15(14), 11240. <https://doi.org/10.3390/su151411240>.

MASSAFRA AND GULLI, 2023

Massafra A and Gulli R (2023) Enabling Bidirectional Interoperability between BIM and BPS through Lightweight Topological Models. Presented at the eCAADe 2023: Digital Design Reconsidered, Graz, Austria, 187–196. <https://doi.org/10.52842/conf.ecaade.2023.2.187>.

MASSAFRA, 2024

Massafra A (2024) *Towards Topology-Oriented Digital Twins for Built Heritage Performance-Based Management* (PhD thesis). Alma Mater Studiorum Università di Bologna. Retrieved from <https://amsdottorato.unibo.it/id/eprint/11502>

MASSAFRA ET AL., 2024a

Massafra A, Coraglia UM, Predari G and Gulli R (2024) Graph-Based Digital Decision Support Systems: Introducing BTwin, a Toolkit for Building Performance Management. Presented at the 2024 European Conference on Computing in Construction. <https://doi.org/10.35490/EC3.2024.242>.

MASSAFRA ET AL., 2024b

Massafra A, Coraglia U, Predari G and Gulli R (2024) Building Information Model Analysis Through Large Language Models and Knowledge Graphs. Presented at the eCAADe 2024: Data-Driven Intelligence, Nicosia, 685–694. <https://doi.org/10.52842/conf.ecaade.2024.1.685>.

MASSAFRA ET AL., 2024c

Massafra A, Jabi W and Gulli R (2024) Topological BIM for building performance management. *Automation in Construction* 166, 105628. <https://doi.org/10.1016/j.autcon.2024.105628>.

MASSAFRA ET AL., 2025a

Massafra A, Predari G, Gulli R, Villa R, Salvalai G, Grecchi M, Bernardini G, Romano G and D’Orazio M (2025) Topological BIM for Occupancy-based Integrated Energy and Safety Analysis of Higher Education Buildings. Presented at the 2025 European Conference on Computing in Construction. <https://doi.org/10.35490/EC3.2025.205>.

MASSAFRA ET AL., 2025b

Massafra A, Al-Harasis DH, Stefanini L and Jabi W (2025) Semi-Automated Dataset Generation for Residential Buildings Using Graph-Based Topological Modelling. *Buildings* 15(8), 1283. <https://doi.org/10.3390/buildings15081283>.

MATSUDA AND OOKA, 2020

Matsuda Y and Ooka R (2020) DEVELOPMENT OF THE DIGITAL-TWIN FOR BUILDING FACILITIES (PART 1): VERIFICATION OF PREDICTIVE ACCURACY OF ANN MODELS FOR HEAT SOURCE SYSTEM BASED ON OPERATION DATA. *Journal of Environmental Engineering (Transactions of AIJ)* 85(770), 267–275. <https://doi.org/10.3130/aije.85.267>.

MATSUNOBU AND COIMBRA, 2023

Matsunobu LM and Coimbra CFM (2023) Reimagining the academic calendar for a changing climate: Modeled impact of shifting the fall term at the University of California. *Cleaner Environmental Systems* 9, 100113. <https://doi.org/10.1016/j.cesys.2023.100113>.

MEMON ET AL., 2025

Memon SA, Shehata W, Rowlinson S and Sunindijo RY (2025) Generative Artificial Intelligence in Architecture, Engineering, Construction, and Operations: A Systematic Review. *Buildings* 15(13), 2270. <https://doi.org/10.3390/buildings15132270>.

MERINO ET AL., 2023

Merino J, Xie X, Moretti N, Chang JY and Parlikad A (2023) Data integration for digital twins in the built environment based on federated data models. *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction* 176(4), 194–211. <https://doi.org/10.1680/jsmic.23.00002>.

MERINO ET AL., 2022

Merino J, Xie X, Moretti N, Chang JY and Parlikad AK (2022) Data integration for digital twins in the built environment. Presented at the 2022 European Conference on Computing in Construction. <https://doi.org/10.35490/EC3.2022.172>.

MILLER, 1998

Miller E (1998) An Introduction to the Resource Description Framework. *Bulletin of the American Society for Information Science and Technology* 25(1), 15–19. <https://doi.org/10.1002/bult.105>.

MOHAMED AND VAN, 2024

Mohamed AA and Van Der Laag Yamu C (2024) Space Syntax has Come of Age: A Bibliometric Review from 1976 to 2023. *Journal of Planning Literature* 39(2), 203–217. <https://doi.org/10.1177/08854122231208018>.

MORETTI ET AL., 2023

Moretti N, Xie X, Merino Garcia J, Chang J and Kumar Parlikad A (2023) Federated Data Modeling for Built Environment Digital Twins. *Journal of Computing in Civil Engineering* 37(4), 04023013. <https://doi.org/10.1061/JCCEE5.CPENG-4859>.

MOSTEIRO-ROMERO ET AL., 2023

Mosteiro-Romero M, Miller C, Chong A and Stouffs R (2023) Elastic buildings: Calibrated district-scale simulation of occupant-flexible campus operation for hybrid work optimization. *Building and Environment* 237, 110318. <https://doi.org/10.1016/j.buildenv.2023.110318>.

NATIONAL, N.D.

National Research Foundation Singapore (n.d.) Virtual Singapore. <https://www.nrf.gov.sg/programmes/virtual-singapore>

NAUATA ET AL., 2020

Nauata N, Chang K-H, Cheng C-Y, Mori G and Furukawa Y (2020) House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation. Available at: <https://github.com/ennauata/housegan> (Accessed 5 March 2026).

NEO4J, 2025A

Neo4j (2025) Model Context Protocol (MCP) integrations for the Neo4j graph database. <https://neo4j.com/developer/genai-ecosystem/model-context-protocol-mcp/>

NEO4J, 2025B

Neo4j Inc. (2025) Neo4j Graph Database & Analytics. <https://neo4j.com/> (accessed 22 October 2025)

NETWORKX, 2025

NetworkX developers (2025) NetworkX - NetworkX Documentation. <https://networkx.org/> (accessed 2 November 2025)

NORTHUMBRIA, 2019

Northumbria University (2019) Visual Newcastle Gateshead. <https://www.northumbria.ac.uk/about-us/academic-departments/architecture-and-built-environment/research/virtual-reality-visualisation/research-enterprise-projects/virtual-newcastle-gateshead/> (accessed 17 January 2024)

NUMPY, 2025

NumPy Team (2025) NumPy. <https://numpy.org/> (accessed 2 November 2025)

ONEM2M, 2019

oneM2M (2019) Base Ontology. <https://git.onem2m.org/MAS/BaseOntology> (accessed 26 October 2025)

PAN ET AL., 2023

Pan S, Luo L, Wang Y, Chen C, Wang J and Wu X (2023) Unifying Large Language Models and Knowledge Graphs: A Roadmap. (Version 3). <https://doi.org/10.48550/ARXIV.2306.08302>.

PANDAS, 2025

pandas (2025) pandas - Python Data Analysis Library. <https://pandas.pydata.org/> (accessed 2 November 2025)

PATEL ET AL., 2023

Patel D, Estevam Schmiedt J, Röger M and Hoffschmidt B (2023) A Model Calibration Approach to U-Value Measurements with Thermography. *Buildings* 13(9), 2253. <https://doi.org/10.3390/buildings13092253>.

PAUWELS, CORRY AND O'DONNELL, 2014

Pauwels P, Corry E and O'Donnell J (2014) Representing SimModel in the Web Ontology Language. In *Computing in Civil and Building Engineering (2014)*. Orlando, Florida, United States: American Society of Civil Engineers, 2271–2278. <https://doi.org/10.1061/9780784413616.282>.

PAUWELS, COSTIN AND RASMUSSEN, 2022

Pauwels P, Costin A and Rasmussen MH (2022) Knowledge Graphs and Linked Data for the Built Environment. In Bolpagni M, Gavina R, and Ribeiro D (eds), *Industry 4.0 for the Built Environment*, Vol. 20. Cham: Springer International Publishing, 157–183. https://doi.org/10.1007/978-3-030-82430-3_7.

PAUWELS AND ROXIN, 2016

Pauwels P and Roxin A (2016) SimpleBIM : from full ifcOWL graphs to simplified building graphs. In *Ework and Ebusiness in architecture, engineering and Construction*. Limassol, Cyprus: CRC Press, 11–18.

PAUWELS AND TERKAJ, 2019

Pauwels P and Terkaj W (2019) ifcOWL Ontology (IFC4_ADD2_TC1). https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD2_TC1/OWL/index.html (accessed 26 October 2025)

PENG, AU-YONG AND MYEDA, 2024

Peng Y, Au-Yong CP and Myeda NE (2024) Knowledge graph of building information modelling (BIM) for facilities management (FM). *Automation in Construction* 165, 105492. <https://doi.org/10.1016/j.autcon.2024.105492>.

PEREZ AND GERMON, 2016

Perez C and Germon R (2016) Graph Creation and Analysis for Linking Actors: Application to Social Data. In Layton R and Watters PA (eds), *Automating Open Source Intelligence: Algorithms for OSINT*, 1st edn. Elsevier, 103–129. <https://doi.org/10.1016/B978-0-12-802916-9.00007-5>.

PETRI ET AL., 2023

Petri I, Rezgui Y, Ghoroghi A and Alzahrani A (2023) Digital twins for performance management in the built environment. *Journal of Industrial Information Integration* 33, 100445. <https://doi.org/10.1016/j.jii.2023.100445>.

PHILIP, 2020

Philip S (2020) Welcome to eppy's documentation! <https://eppy.readthedocs.io/en/latest/index.html> (accessed 2 November 2025)

PIANIGIANI, 2022

Pianigiani O (2022) *Vocabolario Etimologico Della Lingua Italiana*, Vol. 2. Legare Street Press.

PLOTLY, 2025

Plotly (2025) Data Apps for Production | Poltly. <https://plotly.com/> (accessed 2 November 2025)

POPPER AND POPPER, 2008

Popper KR and Popper KR (2008) *The Logic of scientific discovery*, Repr. 2008 (twice). London: Routledge.

PORT, N.D.

Port of Antwerp Bruges (n.d.) Smart port. <https://www.portofantwerpbruges.com/en/our-port/port-future/smart-port> (accessed 17 January 2024)

PREDARI, PRATI AND MASSAFRA, 2021

Predari G, Prati D and Massafra A (2021) Modern Construction in Bologna. The Faculty of Engineering by Giuseppe Vaccaro, 1932–1935. In *Digital Modernism Heritage Lexicon*. Cham, Svizzera: Springer Nature Switzerland AG, 233–258.

PREDARI, PRATI AND MASSAFRA, 2022

Predari G, Prati D and Massafra A (2022) Modern Construction in Bologna. The Faculty of Engineering by Giuseppe Vaccaro, 1932–1935. In Bartolomei C, Ippolito A, and Vizioli SHT (eds), *Digital Modernism Heritage Lexicon*. Cham: Springer International Publishing, 233–258. https://doi.org/10.1007/978-3-030-76239-1_11.

QUEEN'S, QUEEN'S AND 2018., N.D.

Queen's University, Queen's University Greenhouse Gas Inventory Report 2017, 2018. (n.d.).

RAJ, 2025

Raj P (2025) *Applied Graph Data Science: Graph Algorithms and Platforms, Knowledge Graphs, Neural Networks, and Applied Use Cases*, 1st ed. Chantilly: Elsevier Science & Technology.

RAMONELL, CHACÓN AND POSADA, 2023

Ramonell C, Chacón R and Posada H (2023) Knowledge graph-based data integration system for digital twins of built assets. *Automation in Construction* 156, 105109. <https://doi.org/10.1016/j.autcon.2023.105109>.

RANE, CHOUDHARY AND RANE, 2023

Rane N, Choudhary S and Rane J (2023) Integrating Building Information Modelling (BIM) with ChatGPT, Bard, and similar generative artificial intelligence in the architecture, engineering, and construction industry: applications, a novel framework, challenges, and future scope. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.4645601>.

RASMUSSEN ET AL., 2019

Rasmussen MH, Lefrançois M, Pauwels P, Hviid CA and Karlshøj J (2019) Managing interrelated project information in AEC Knowledge Graphs. *Automation in Construction* 108, 102956. <https://doi.org/10.1016/j.autcon.2019.102956>.

RASMUSSEN ET AL., 2017

Rasmussen MH, Pauwels P, Hviid CA and Karlshøj J (2017) Proposing a Central AEC Ontology That Allows for Domain Specific Extensions. In *Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction*. Heraklion, Crete, Greece: Heriot-Watt University, 237–244. <https://doi.org/10.24928/JC3-2017/0153>.

REINISCH, KOFLER AND KASTNER, 2010

Reinisch C, Kofler MJ and Kastner W (2010) ThinkHome: A smart home as digital ecosystem. In *4th IEEE International Conference on Digital Ecosystems and Technologies*. Dubai, United Arab Emirates: IEEE, 256–261. <https://doi.org/10.1109/DEST.2010.5610636>.

REN ET AL., 2023

Ren J, Xia F, Lee I, Noori Hoshyar A and Aggarwal C (2023) Graph Learning for Anomaly Analytics: Algorithms, Applications, and Challenges. *ACM Transactions on Intelligent Systems and Technology* 14(2), 1–29. <https://doi.org/10.1145/3570906>.

ROBERT, 1993

Robert McNeel & Associates (1993, 2025) Rhino - Rhinoceros 3D. <https://www.rhino3d.com/it/> (accessed 6 November 2025)

RUOHOMAKI ET AL., 2018

Ruohomaki T, Airaksinen E, Huuska P, Kesaniemi O, Martikka M and Suomisto J (2018) Smart City Platform Enabling Digital Twin. In *2018 International Conference on Intelligent Systems (IS)*. Funchal - Madeira, Portugal: IEEE, 155–161. <https://doi.org/10.1109/IS.2018.8710517>.

RWTH-E3D, N.D.

RWTH-E3D (n.d.) TUBES System Ontology. <https://rwth-e3d.github.io/tso/> (accessed 26 October 2025)

SAHA, BORGEFORS AND SANNITI, 2017

Saha PK, Borgfors G and Sanniti di Baja G (2017) Skeletonization and its applications – a review. In *Skeletonization*. Elsevier, 3–42. <https://doi.org/10.1016/B978-0-08-101291-8.00002-X>.

SCHROTTER AND HÜRZELER, 2020

Schrotter G and Hürzeler C (2020) The Digital Twin of the City of Zurich for Urban Planning. *PGF – Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88(1), 99–112. <https://doi.org/10.1007/s41064-020-00092-2>.

SEMPRINI ET AL., 2016

Semprini G, Marinosci C, Ferrante A, Predari G, Mochi G, Garai M and Gulli R (2016) Energy management in public institutional and educational buildings: The case of the school of engineering and architecture in Bologna. *Energy and Buildings* 126, 365–374. <https://doi.org/10.1016/j.enbuild.2016.05.009>.

SHARMILA AND KAVITHA, 2023

Sharmila G and Kavitha Devi MK (2023) BTLA-LSDG: Blockchain-Based Triune Layered Architecture for Authenticated Subgraph Query Search in Large-Scale Dynamic Graphs. *IETE Journal of Research* 1–24. <https://doi.org/10.1080/03772063.2023.2171910>.

SINGHAL, 2012

Singhal A (2012) Introducing the Knowledge Graph: things, not strings. <https://blog.google/products/search/introducing-knowledge-graph-things-not/> (accessed 20 September 2025)

ŠKRLJ ET AL., 2025

Škrlj B, Koloski B, Pollak S and Lavrač N (2025) From Symbolic to Neural and Back: Exploring Knowledge Graph-Large Language Model Synergies. (Version 1). arXiv. <https://doi.org/10.48550/ARXIV.2506.09566>.

SQLITE, 2025

SQLite Consortium (2025) SQLite Home Page. <https://sqlite.org/> (accessed 2 November 2025)

STAVROPOULOS ET AL., 2012

Stavropoulos TG, Vrakas D, Vlachava D and Bassiliades N (2012) BOnSAI: a smart building ontology for ambient intelligence. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*. Craiova Romania: ACM, 1–12. <https://doi.org/10.1145/2254129.2254166>.

SUCCAR AND POIRIER, 2020

Succar B and Poirier E (2020) Lifecycle information transformation and exchange for delivering and managing digital and physical assets. *Automation in Construction* 112, 103090. <https://doi.org/10.1016/j.autcon.2020.103090>.

TAKVA, TAKVA AND ÇAKICI, 2025

Takva Ç, Takva FG and Çakıcı FZ (2025) Thermal Analysis of the Building Envelope with Infrared Thermography and Simulation in Educational Buildings in the Cold Climate Region. *Buildings* 15(11), 1759. <https://doi.org/10.3390/buildings15111759>.

TAO, ZHANG AND NEE, 2019

Tao F, Zhang M and Nee AYC (2019) *Digital twin driven smart manufacturing*. London: Academic press.

TARDY, 2025

Tardy F (2025) Methodology for estimating building thermal resistance and heat capacity values in-situ using exterior measurements and meteorological data. *Journal of Building Engineering* 101, 111768. <https://doi.org/10.1016/j.jobe.2025.111768>.

TECLAW ET AL., 2024

Teclaw W, O'Donnel J, Kukkonen V, Pauwels P, Labonnote N and Hjelseth E (2024) Federating cross-domain BIM-based knowledge graph. *Advanced Engineering Informatics* 62, 102770. <https://doi.org/10.1016/j.aei.2024.102770>.

TRAN ET AL., 2021

Tran H, Nguyen TN, Christopher P, Bui D-K, Khoshelham K and Ngo TD (2021) A digital twin approach for geometric quality assessment of as-built prefabricated façades. *Journal of Building Engineering* 41, 102377. <https://doi.org/10.1016/j.jobe.2021.102377>.

UHM ET AL., 2025

Uhm M, Kim J, Ahn S, Jeong H and Kim H (2025) Effectiveness of retrieval augmented generation-based large language models for generating construction safety information. *Automation in Construction* 170, 105926. <https://doi.org/10.1016/j.autcon.2024.105926>.

UNIVERSITY, CARBON AND 2017., N.D.

University of West England, Carbon Management Plan 2013-2020 (Version: 03), 2017. (n.d.).

W3C, 2021

W3C (2021) Building Topology Ontology. <https://w3c-lbd-cg.github.io/bot/>

WADUD, ROYSTON AND SELBY, 2019

Wadud Z, Royston S and Selby J (2019) Modelling energy demand from higher education institutions: A case study of the UK. *Applied Energy* 233–234, 816–826. <https://doi.org/10.1016/j.apenergy.2018.09.203>.

WANG ET AL., 2024

Wang M, Lilis GN, Mavrokapnidis D, Katsigarakis K, Korolija I and Rovas D (2024) A knowledge graph-based framework to automate the generation of building energy models using geometric relation checking and HVAC topology establishment. *Energy and Buildings* 325, 115035. <https://doi.org/10.1016/j.enbuild.2024.115035>.

WANG ET AL., 2022

Wang Q, Zhou G, Song R, Xie Y, Luo M and Yue T (2022) Continuous space ant colony algorithm for automatic selection of orthophoto mosaic seamline network. *ISPRS Journal of Photogrammetry and Remote Sensing* 186, 201–217. <https://doi.org/10.1016/j.isprsjprs.2022.02.011>.

WANG AND SACKS, 2025

Wang Z and Sacks R (2025) Building Information Graphs (BIGs): remodeling building information for learning and applications. *Data-Centric Engineering* 6, e44. <https://doi.org/10.1017/dce.2025.10024>.

WIKIMEDIA, 2025

Wikimedia Foundation (n.d.) Wikidata. https://www.wikidata.org/wiki/Wikidata:Main_Page (accessed 22 October 2025)

WU ET AL., 2022

Wu L, Cui P, Pei J and Zhao L (eds) (2022) *Graph neural networks: foundations, frontiers, and applications*. Singapore: Springer.

WURZER AND LORENZ, 2016

Wurzer G and Lorenz WE (2016) SpaceBook: A Case Study of Social Network Analysis in Adjacency Graphs. In Hernejoja A, Österlund T, and Markkanen P (eds), *Complexity & Simplicity: Proceedings of the 34th eCAADe Conference, Volume 2*. Oulu, Finland, 229–238. <https://hdl.handle.net/20.500.12708/64548>

XIE ET AL., 2021

Xie X, Moretti N, Merino J and Parlikad AK (2021) Ontology-Based Spatial and System Hierarchies Federation for Fine-Grained Building Energy Analysis. Presented at the 38th International Conference of CIB W78, Luxembourg.

ŽÁČEK AND JANOŠEK, 2017

Žáček M and Janošek M (2017) SBonto: ONTOLOGY OF SMART BUILDING. *Far East Journal of Electronics and Communications* 17(5), 1101–1109. <https://doi.org/10.17654/EC017051101>.

ZHANG ET AL., 2025

Zhang X, Zheng J, Li P, Yang Y, Ye Y, Causone F and Shi X (2025) A review of building digital twins: Framework and enabling technologies. *Journal of Building Engineering* 111, 113117. <https://doi.org/10.1016/j.jobe.2025.113117>.

GLOSSARY

AECO – Architecture, Engineering, Construction, and Operations
AHU – Air Handling Unit
AI – Artificial Intelligence
ANN – Artificial Neural Network
API – Application Programming Interface
BEM – Building Energy Modeling
BI – Business Intelligence
BIM – Building Information Modeling
BMS – Building Management System
BOT – Building Topology Ontology
BPS – Building Performance Simulation
BSM – Building Safety Model
CSV – Comma-Separated Values
CO₂E – Carbon Dioxide Equivalent Emissions
DT – Digital Twin
DSS – Decision Support System
DTS – Deemed-To-Satisfy
EBDB – Energy Bill Database
EC – Energy Cost
EP – EnergyPlus
EWI – Energy What-If
FDD – Fault Detection and Diagnostics
FDM – Federated Data Modelling
FM – Facility Management
FOF – Federated Ontology Framework
FSC – Fire Safety Code
GbXML – Green Building XML (Green Building Markup Language)
GDS – Graph Data Science
GNN – Graph Neural Network
GQL – Graph Query Language
GUI – Graphical User Interface
HVAC – Heating, Ventilation, and Air Conditioning
IAQ – Indoor Air Quality
IEDB – Indoor Environment Database
IFC – Industry Foundation Classes

IoT – Internet of Things
IT – Information Technology
JSON – JavaScript Object Notation
JSON-LD – JSON for Linked Data
KG – Knowledge Graph
KPIO – Key Performance Indicator Ontology
LLM – Large Language Model
LBD – Linked Building Data
LD – Linked Data
LOD – Level of Development
LPG – Labelled Property Graph
MCP – Model Context Protocol
MEP – Mechanical, Electrical, and Plumbing
ML – Machine Learning
MSE – Mean Squared Error
OWL – Web Ontology Language
RAG – Retrieval-Augmented Generation
RDBMS – Relational Database Management System
RDF – Resource Description Framework
RMSE – Root Mean Squared Error
RF – Random Forest
SOSA – Sensors, Observations, Samples, and Actuators
SQL – Structured Query Language
SSN – Semantic Sensor Network
SWI – Safety What-If
TBIM – Topological BIM
UCDC – Uniform Component Decomposition Centrality
UID – Unique Identifier
URI – Uniform Resource Identifier
VPL – Visual Programming Language
W3C – World Wide Web Consortium

ACKNOWLEDGEMENTS

The author wishes to thank all those who contributed to this project, particularly with regard to the demonstrative section.

For their contribution to the research presented in Chapters 3 and 4, conducted as part of the BETWIN project (AlmaValue 2023 Call, Università di Bologna), thanks are due to Ugo Maria Coraglia, Giorgia Predari, and Riccardo Gulli.

For their contribution to the research presented in Chapter 5, the author expresses gratitude to Wassim Jabi, Dania Al-Harasis, and Lorenzo Stefanini.

For their contribution to the research presented in Chapters 6 and 9, conducted as part of the DIGITMAN project (PRIN 2022 programme, ID 2022JMRX2A), the author wishes to thank Marco D’Orazio, Gabriele Berardini, Guido Romano, Graziano Salvalai, Manuela Grecchi, Roberto Villa, Giorgia Predari and Riccardo Gulli.

For their contribution to the research presented in Chapter 8, thanks are extended to Ugo Maria Coraglia, Silvia Di Turi, Domenico Palladino, and Michele Reginaldi.

For their contribution to the supervision of the research conducted in Chapter 8, the author wishes to thank Riccardo Gulli and Wassim Jabi.

More broadly, the author wishes to express sincere gratitude to the wider community of developers and AECO professionals whose contributions to the open-source libraries referenced throughout this work have been fundamental to the research. Equal appreciation is extended to the developers and maintainers of the reference ontologies forming the federated framework proposed in this study, whose work has provided an essential foundation for this research.

Editorial note: AI-assisted tools were employed to support linguistic and stylistic revision throughout the book. All scientific contents, arguments, and responsibilities remain entirely with the author.

Finito di stampare nel mese di marzo 2026
per i tipi di Bologna University Press